数值试验二 赵衍智 **2020511041**

试验一**:** 体会数值方法的不稳定性对数值结果的影响

1. 显式欧拉法

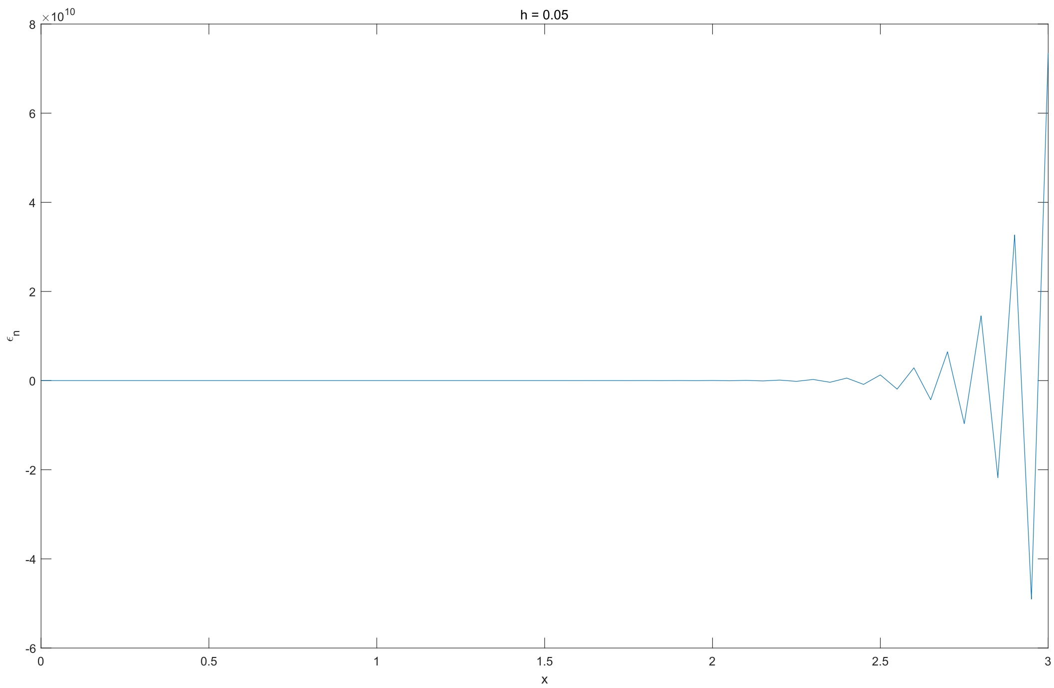
(1). 理论分析

$$
\begin{aligned}
&\because y_{n+1} = y_n + hf(x_n, y_n) = y_n - 50hy_n = (1-50h)y_n \\
&\text{假设}y_n\text{有误差}\epsilon_n,\text{即实际计算值}\bar{y}_n = y_n + \epsilon_n \\
&\therefore \bar{y}_{n+1} = (1-50h)\bar{y}_n = (1-50h)(y_n + \epsilon_n) \\
&\therefore y_{n+1}\text{的误差}\epsilon_{n+1} = \bar{y}_{n+1} - y_{n+1} = (1-50h)\epsilon_n \\
&\therefore \text{迭代格式具有稳定性} \iff \frac{|\epsilon_{n+1}|}{|\epsilon_n|} \le 1 \iff |1-50h| \le 1 \iff 0 < h \le 0.04
\end{aligned}
$$

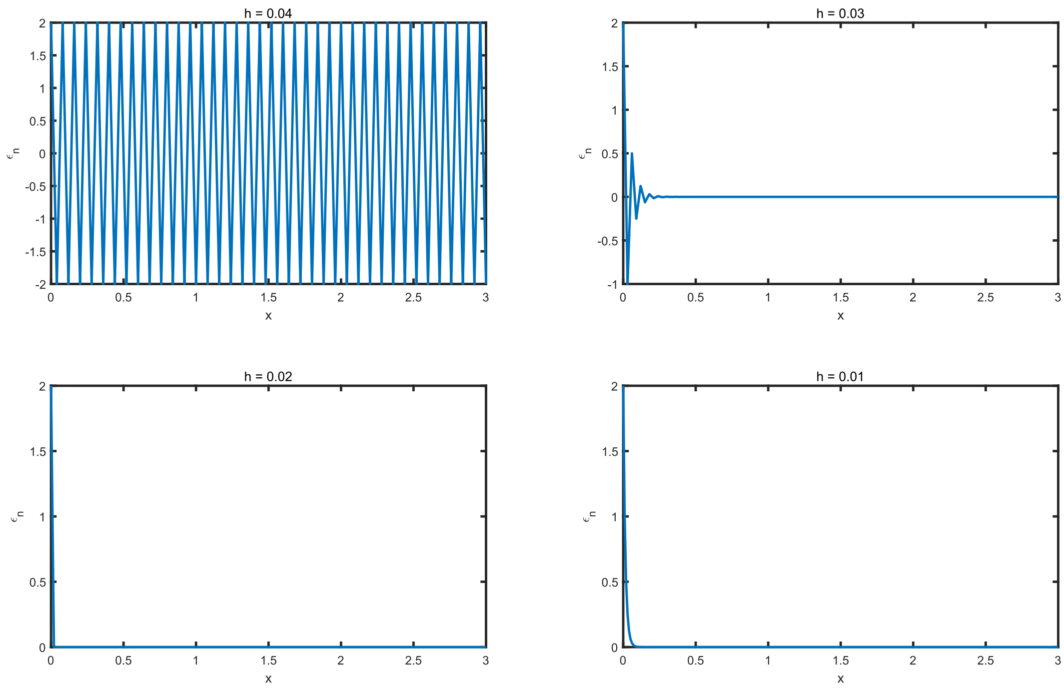
(2). 试验结果

在数值试验中一律取$y_0 = y(0) = 100, \bar{y}_0 = 102$

- $h = 0.05$, 试验区间取为$[0, 6]$



- $h = 0.04, 0.03, 0.02, 0.01$; 试验区间取为$[0, 3]$



(3). 结果分析

$$(i)\ h = 0.05\text{时误差}\epsilon_n\text{不断扩大}$$

$$(ii)\ h = 0.04, 0.03, 0.02, 0.01\text{时迭代格式稳定}$$

$$(iii)\ h = 0.04\text{时}\epsilon_n\text{在}2\text{和}-2\text{处来回波动, 因为}\frac{\epsilon_{n+1}}{\epsilon_n} = 1 - 50h = -1, n = 0, 1, 2, \cdots$$

$$(iv)\ h = 0.02\text{时}\epsilon_n\text{衰减最快, 在}x_1\text{处衰减为零, 因为}\frac{|\epsilon_{n+1}|}{|\epsilon_n|} = |1 - 50h| = 0, n = 0, 1, 2, \cdots$$

(4). 代码

- 以$h = 0.04$为例：

```
clf
y0 = 100;
y0_ = 102;
h = 0.04; % 步长
l = 3; % 区间长度
m = l/h + 1;
x = 0:h:l;
Y = zeros(1,m);
Y_ = zeros(1,m);
Y(1) = y0;
Y_(1) = y0_;
for i = 2:1:m
    Y(i) = (1 - 50*h)*Y(i-1);
    Y_(i) = (1 - 50*h)*Y_(i-1);
end
epsilon = Y_ - Y;
subplot(2,2,1);
plot(x,epsilon);
xlabel('x');
ylabel('\epsilon_n');
title('h = 0.04');
```

2. 隐式欧拉法

(1). 理论分析

$$\because y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}) = y_n - 50hy_{n+1}, \text{即} y_{n+1} = \frac{y_n}{1+50h}$$

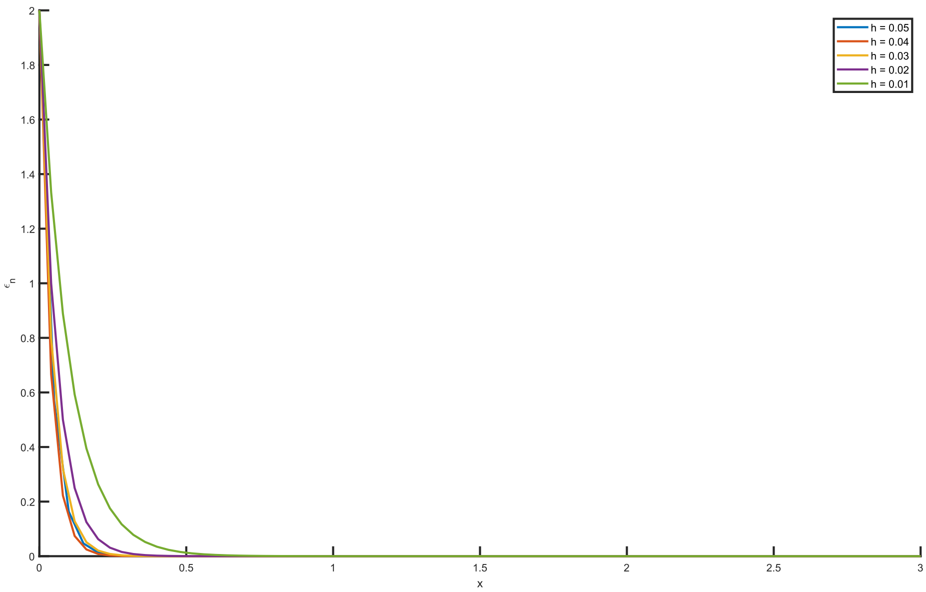
$$\text{设} y_n \text{有误差} \epsilon_n, \text{则实际计算值} \bar{y}_n = y_n + \epsilon_n$$

$$\therefore \bar{y}_{n+1} = \frac{\bar{y}_n}{1+50h}, \quad y_{n+1}\text{有误差} \epsilon_{n+1} = \bar{y}_{n+1} - y_n = \frac{\epsilon_n}{1+50h}$$

$$\therefore \text{迭代格式具有稳定性} \iff \frac{|\epsilon_{n+1}|}{|\epsilon_n|} \le 1 \iff \frac{1}{|1+50h|} \le 1 \iff h > 0$$

(2). 试验结果

- h取$0.05, 0.04, 0.03, 0.02, 0.01$; 试验区间取为$[0, 3]$
  迭代格式均稳定



(3). 结果分析

$$\text{随着} h \text{的增大}, |\epsilon_n| \text{衰减的速率增大}, \text{因为} \frac{|\epsilon_{n+1}|}{|\epsilon_n|} = \frac{1}{|1+50h|}; \ h\text{增大}, \frac{|\epsilon_{n+1}|}{|\epsilon_n|} = \frac{1}{|1+50h|} \text{减小}, \epsilon_n \text{衰减加快}$$

(4). 代码

- 以$h = 0.05, 0.04$为例：

```
clf
y0 = 100;
y0_ = 102;

hold on
h1 = 0.05; % 步长
l = 3; % 区间长度
m = l/h1 + 1;
x = 0:h1:l;
Y = zeros(1,m);
Y_ = zeros(1,m);
Y(1) = y0;
Y_(1) = y0_;
for i = 2:1:m
    Y(i) = Y(i-1)/(1 + 50*h1);
    Y_(i) = Y_(i-1)/(1 + 50*h1);
end
epsilon = Y_ - Y;
plot(x,epsilon);
xlabel('x');
ylabel('\epsilon_n');

h2 = 0.04;
m = l/h2 + 1;
x = 0:h2:l;
Y = zeros(1,m);
Y_ = zeros(1,m);
Y(1) = y0;
Y_(1) = y0_;
for i = 2:1:m
    Y(i) = Y(i-1)/(1 + 50*h2);
    Y_(i) = Y_(i-1)/(1 + 50*h2);
end
epsilon = Y_ - Y;
plot(x,epsilon);
```

试验二：体会算法的保结构性

1. 梯形法

(1). 理论分析

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} + \frac{h}{2}[\begin{pmatrix} x'_n \\ y'_n \end{pmatrix} + \begin{pmatrix} x'_{n+1} \\ y'_{n+1} \end{pmatrix}] = \begin{pmatrix} x_n \\ y_n \end{pmatrix} + \frac{h}{2}[\begin{pmatrix} \frac{9}{2}y_n \\ -2x_n \end{pmatrix} + \begin{pmatrix} \frac{9}{2}y_{n+1} \\ -2x_{n+1} \end{pmatrix}]$$

$$即 \begin{cases} x_{n+1} - \dfrac{9h}{4}y_{n+1} = x_n + \dfrac{9h}{4}y_n \\ y_{n+1} + hx_{n+1} = y_n - hx_n \end{cases}$$

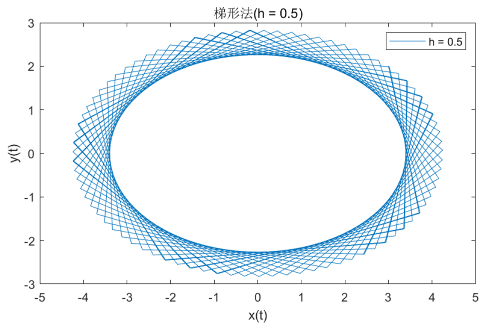
$$即 \begin{cases} y_{n+1} = -\dfrac{8h}{4+9h^2}x_n + \dfrac{4-9h^2}{4+9h^2}y_n \\ x_{n+1} = \dfrac{4-9h^2}{4+9h^2}x_n + \dfrac{18h}{4+9h^2}y_n \end{cases}$$

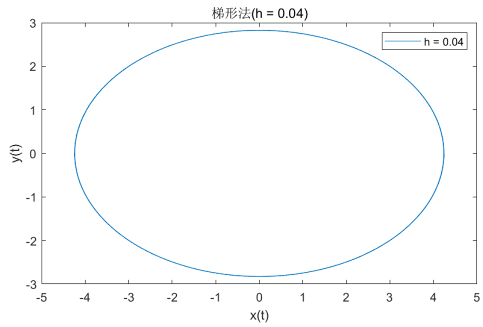
$$\therefore x_{n+1}^2 + \frac{9}{4}y_{n+1}^2 = x_n^2 + \frac{9}{4}y_n^2$$

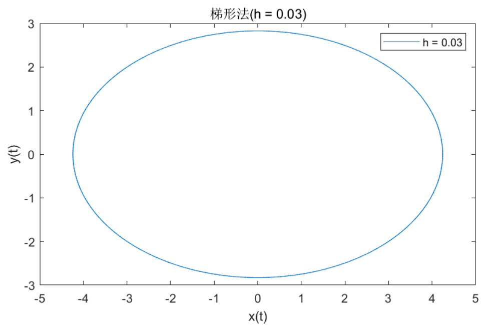
$$\therefore x_n^2 + \frac{9}{4}y_n^2 = x_0^2 + \frac{9}{4}y_0^2 = 18$$

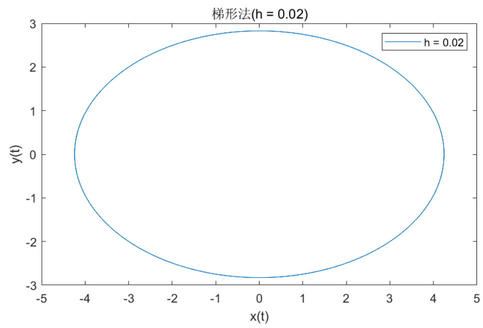
(2). 试验结果

- $h = 0.5$时$n = 1, 2, \cdots, 100$; $h = 0.04, 0.03, 0.02$时$n = 0, 1, 2, \cdots, 2000$









(3). 结果分析

梯形法相图中的每个点都在椭圆$x^2 + \frac{9}{4}y^2 = 18$上，梯形法具有保结构性

(4). 代码

- 以$h = 0.5$为例

```
clf
x_0 = 3;
y_0 = 2;

subplot(2,2,1);
h = 0.5;
n = 100;
X = zeros(1,n);
Y = X;
X(1) = x_0;
Y(1) = y_0;
for i = 2:1:n
    X(i) = (4 - 9*h^2)*X(i-1)/(4+9*h^2) + 18*h*Y(i-1)/(4 + 9*h^2);
    Y(i) = -8*h*X(i-1)/(4 + 9*h^2) + (4 - 9*h^2)*Y(i-1)/(4+9*h^2);
end
plot(X,Y);
xlabel('x(t)');
ylabel('y(t)');
legend('h = 0.5')
title('梯形法(h = 0.5)')
```

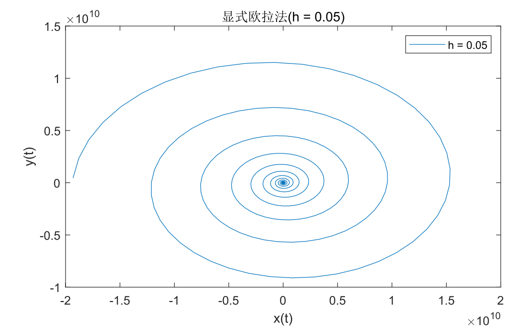
2. 显式欧拉法

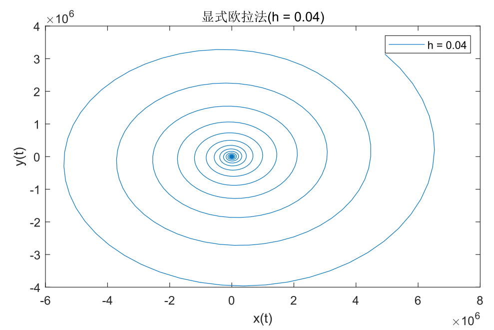
(1). 理论分析

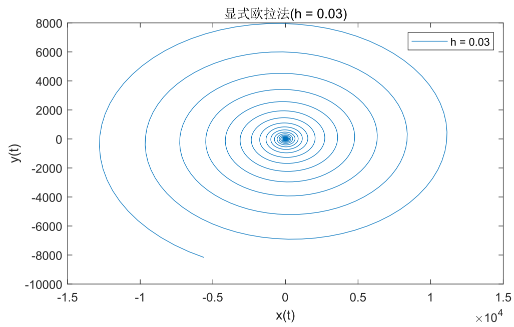
$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} + h \begin{pmatrix} x_n' \\ y_n' \end{pmatrix} = \begin{pmatrix} x_n + \frac{9h}{2} y_n \\ y_n - 2hx_n \end{pmatrix}$$

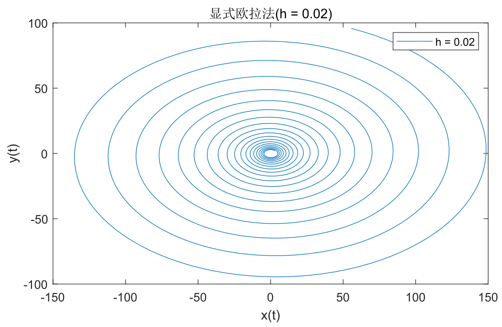
(2). 试验结果

- $n = 0, 1, 2, \cdots, 2000$









(3). 结果分析

显式欧拉法相图中的点不在固定的轨迹上，改变h和n，点的轨迹随之改变。显式欧拉法不具有保结构性

(4). 代码

- 以$h = 0.05$为例：

```
clf
x_0 = 3;
y_0 = 2;

%% 显式
subplot(2,2,1);
h = 0.05;
n = 2000;
X = zeros(1,n);
Y = X;
X(1) = x_0;
Y(1) = y_0;
for i = 2:1:n
    X(i) = X(i-1) - 9*h*Y(i-1)/2;
    Y(i) = Y(i-1) + 2*h*X(i-1);
end
plot(X,Y);
legend('h = 0.05');
xlabel('x(t)');
ylabel('y(t)');
title('显式欧拉法(h = 0.05)')
```

3. 隐式欧拉法

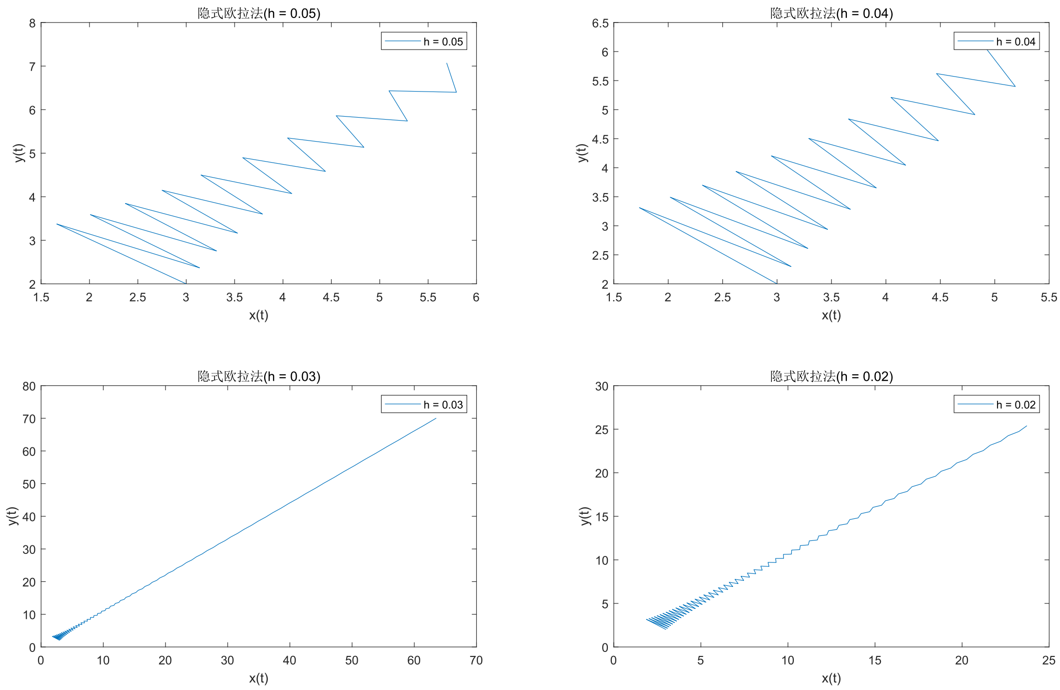
(1). 理论分析

$$\because \begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} + h\begin{pmatrix} x'_{n+1} \\ y'_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} + \begin{pmatrix} \frac{9h}{2} y_{n+1} \\ -2hx_{n+1} \end{pmatrix}$$

$$\therefore \begin{cases} y_{n+1} = -\dfrac{2h}{1+9h^2}x_n + \dfrac{1}{1+9h^2}y_n \\ x_{n+1} = \dfrac{1}{1+9h^2}x_n + \dfrac{9h}{2+18h^2}y_n \end{cases}$$

(2). 试验结果

- $h = 0.05, 0.04$时$n = 0, 1, 2, \cdots, 20$; $h = 0.03, 0.02$时$n = 0, 1, 2, \cdots, 100$



(3). 结果分析

隐式欧拉法相图中的点不在固定的轨迹上，改变h和n，点的轨迹随之改变。隐式欧拉法不具有保结构性

(4). 代码

- 以$h = 0.05$为例：

```
%%% 隐式
subplot(2,2,1);
h = 0.05;
n = 20;
X = zeros(1,n);
Y = X;
X(1) = x_0;
Y(1) = y_0;
for i = 2:1:n
    X(i) = -2*h*X(i-1)/(1+9*h^2) + Y(i-1)/(1 + 9*h^2);
    Y(i) = X(i-1)/(1 + 9*h^2) + 9*h*Y(i-1)/(2+18*h^2);
end
plot(X,Y);
legend('h = 0.05');
xlabel('x(t)');
ylabel('y(t)');
title('隐式欧拉法(h = 0.05)')
```

4. 四阶 Runge-Kutta 法

(1). 理论分析

$$\text{记}x'(t)=f(t,x,y)=\frac{9}{2}y(t),\ y'(t)=g(t,x,y)=-2x(t)$$

$$\text{则四阶Runge-Kutta公式为}:\begin{pmatrix}x_{n+1}\\y_{n+1}\end{pmatrix}=\begin{pmatrix}x_n\\y_n\end{pmatrix}+\frac{h}{6}[\begin{pmatrix}k_1^{(1)}\\k_1^{(2)}\end{pmatrix}+2\begin{pmatrix}k_2^{(1)}\\k_2^{(2)}\end{pmatrix}+2\begin{pmatrix}k_3^{(1)}\\k_3^{(2)}\end{pmatrix}+\begin{pmatrix}k_4^{(1)}\\k_4^{(4)}\end{pmatrix}]$$

$$\because\begin{pmatrix}k_1^{(1)}\\k_1^{(2)}\end{pmatrix}=\begin{pmatrix}f(t_n,x_n,y_n)\\g(t_n,x_n,y_n)\end{pmatrix}=\begin{pmatrix}\frac{9}{2}y_n\\-2x_n\end{pmatrix}$$

$$\begin{pmatrix}k_2^{(1)}\\k_2^{(2)}\end{pmatrix}=\begin{pmatrix}f(t_n+\frac{h}{2},x_n+\frac{h}{2}k_1^{(1)},y_n+\frac{h}{2}k_1^{(2)})\\g(t_n+\frac{h}{2},x_n+\frac{h}{2}k_1^{(1)},y_n+\frac{h}{2}k_1^{(2)})\end{pmatrix}=\begin{pmatrix}\frac{9}{2}(y_n+\frac{h}{2}k_1^{(2)})\\-2(x_n+\frac{h}{2}k_1^{(1)})\end{pmatrix}=\begin{pmatrix}\frac{9}{2}y_n-\frac{9h}{2}x_n\\-2x_n-\frac{9h}{2}y_n\end{pmatrix}$$

$$\begin{pmatrix}k_3^{(1)}\\k_3^{(2)}\end{pmatrix}=\begin{pmatrix}f(t_n+\frac{h}{2},x_n+\frac{h}{2}k_2^{(1)},y_n+\frac{h}{2}k_2^{(2)})\\g(t_n+\frac{h}{2},x_n+\frac{h}{2}k_2^{(1)},y_n+\frac{h}{2}k_2^{(2)})\end{pmatrix}=\begin{pmatrix}\frac{9}{2}(y_n+\frac{h}{2}k_2^{(2)})\\-2(x_n+\frac{h}{2}k_2^{(1)})\end{pmatrix}=\begin{pmatrix}(\frac{9}{2}-\frac{81h^2}{8})y_n-\frac{9h}{2}x_n\\(\frac{9h^2}{2}-2)x_n-\frac{9h}{2}y_n\end{pmatrix}$$

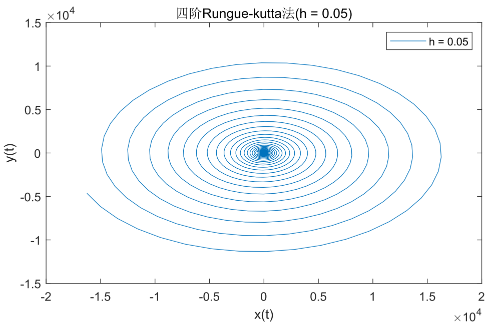
$$\begin{pmatrix}k_4^{(1)}\\k_4^{(2)}\end{pmatrix}=\begin{pmatrix}f(t_n+\frac{h}{2},x_n+\frac{h}{2}k_3^{(1)},y_n+\frac{h}{2}k_3^{(2)})\\g(t_n+\frac{h}{2},x_n+\frac{h}{2}k_3^{(1)},y_n+\frac{h}{2}k_3^{(2)})\end{pmatrix}=\begin{pmatrix}\frac{9}{2}(y_n+\frac{h}{2}k_3^{(2)})\\-2(x_n+\frac{h}{2}k_3^{(1)})\end{pmatrix}=\begin{pmatrix}(\frac{9}{2}-\frac{81h^2}{8})y_n+(\frac{81h^3}{8}-\frac{9h}{2})x_n\\(\frac{81h^3}{8}-\frac{9h}{2})y_n+(\frac{9h^2}{2}-2)x_n\end{pmatrix}$$

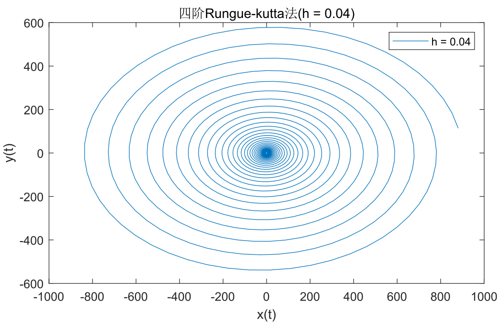
$$\therefore\begin{pmatrix}x_{n+1}\\y_{n+1}\end{pmatrix}=\begin{pmatrix}x_n\\y_n\end{pmatrix}+\frac{h}{6}\begin{pmatrix}(-\frac{243h^2}{8}+27)y_n+(\frac{81h^3}{8}-\frac{45h}{2})x_n\\(\frac{81h^3}{8}-\frac{45h}{4})y_n+(\frac{27h^2}{2}-12)x_n\end{pmatrix}$$

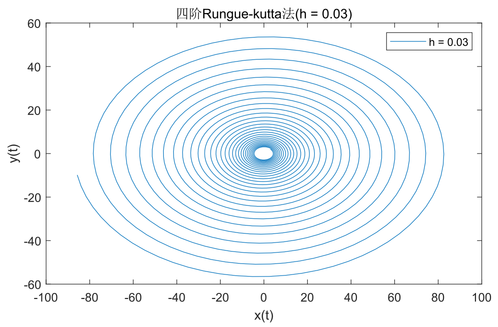
$$\therefore\begin{cases}x_{n+1}=(\frac{27h^4}{16}-\frac{15h^2}{4}+1)x_n+(-\frac{81h^3}{16}+\frac{9h}{2})y_n\\y_{n+1}=(\frac{9h^3}{4}-2h)x_n+(\frac{27h^4}{16}-\frac{15h^2}{8}+1)y_n\end{cases}$$

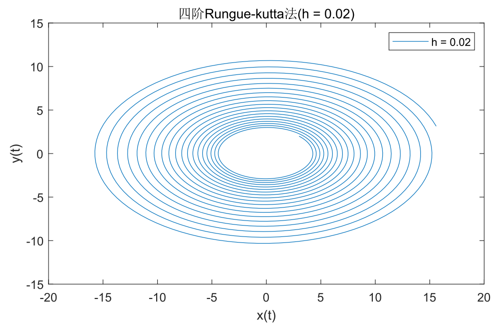
(2). 试验结果

- $n=0,1,2,\cdots,2000$









(3). 结果分析

四阶Runge-Kutta法相图中的点不在固定的轨迹上，改变h和n，点的轨迹随之改变。四阶Runge-Kutta法不具有保结构性

(4). 代码

- 以$h = 0.05, 0.04$为例：

```
clf
x_0 = 3;
y_0 = 2;
h = 0.05;
n = 2000;
X = zeros(1,n);
Y = X;
X(1) = x_0;
Y(1) = y_0;
for i = 2:1:n
    X(i) = (27*h^4/16 - 15*h^2/4 + 1)*X(i-1) + (-81*h^3/16 + 9*h/2)*Y(i-1);
    Y(i) = (9*h^3/4 - 2*h)*X(i-1) + (27*h^4/16 - 15*h^2/8 + 1)*Y(i-1);
end
subplot(2,2,1);
plot(X,Y);
xlabel('x(t)');
ylabel('y(t)');
legend('h = 0.05')
title('四阶Rungue-kutta法(h = 0.05)')

subplot(2,2,2);
h = 0.04;
n = 2000;
X = zeros(1,n);
Y = X;
X(1) = x_0;
Y(1) = y_0;
for i = 2:1:n
    X(i) = (27*h^4/16 - 15*h^2/4 + 1)*X(i-1) + (-81*h^3/16 + 9*h/2)*Y(i-1);
    Y(i) = (9*h^3/4 - 2*h)*X(i-1) + (27*h^4/16 - 15*h^2/8 + 1)*Y(i-1);
end
plot(X,Y);
xlabel('x(t)');
ylabel('y(t)');
legend('h = 0.04')
title('四阶Rungue-kutta法(h = 0.04)')
```

试验三: 体会非线性方程的迭代求解

1. 将二阶方程组化为一阶方程组

$$\text{设}\theta(t)=\theta_1(t),\ \frac{d\theta_1(t)}{dt}=\theta_2(t)$$

$$\therefore \text{原二阶方程组可化为}\begin{cases}\dfrac{d\theta_2(t)}{dt}=-\sin\theta_1,\ 0<t\le 10\\ \dfrac{d\theta_1(t)}{dt}=\theta_2(t)\\ \theta_1(0)=\dfrac{\pi}{3}\\ \theta_2(0)=-\dfrac{1}{2}\end{cases}$$

2. 隐式欧拉法

(1). 理论分析

$$\begin{pmatrix}\theta_{1(n+1)}\\ \theta_{2(n+1)}\end{pmatrix}=\begin{pmatrix}\theta_{1(n)}\\ \theta_{2(n)}\end{pmatrix}+h\begin{pmatrix}\theta'_{1(n+1)}\\ \theta'_{2(n+1)}\end{pmatrix}=\begin{pmatrix}\theta_{1(n)}\\ \theta_{2(n)}\end{pmatrix}+h\begin{pmatrix}\theta_{2(n+1)}\\ -\sin[\theta_{1(n+1)}]\end{pmatrix} \tag{*}$$

对一阶方程组$(*)$进行数值求解有多种求解方法, 本试验共尝试两种方法：

[方法一]: $(*)$可化为 $\begin{cases}\theta_{1(n+1)}+h^2\sin[\theta_{1(n+1)}]=\theta_{1(n)}+h\theta_{2(n)}\\ \theta_{2(n+1)}=\theta_{2(n)}-h\sin[\theta_{1(n+1)}]\end{cases}$

通过Matlab内置函数 vpasolve() 直接计算$\theta_{1(n+1)}$和$\theta_{2(n+1)}$的数值解

[方法二]: $(*)$可化为 $\begin{cases}\theta_{1(n+1)}=\theta_{1(n)}+h\theta_{2(n+1)}\\ \theta_{2(n+1)}=\theta_{2(n)}-h\sin[\theta_{1(n+1)}]\end{cases}$, 迭代求解$\theta_{1(n+1)}$与$\theta_{2(n+1)}$：

先用显式欧拉法求初始迭代值$\theta_{1(n+1)}^{(0)}$和$\theta_{2(n+1)}^{(0)}$, 即$\theta_{1(n+1)}^{(0)}=\theta_{1(n)}+h\theta_{2(n)}, \theta_{2(n+1)}^{(0)}=\theta_{2(n)}-h\sin[\theta_{1(n)}]$

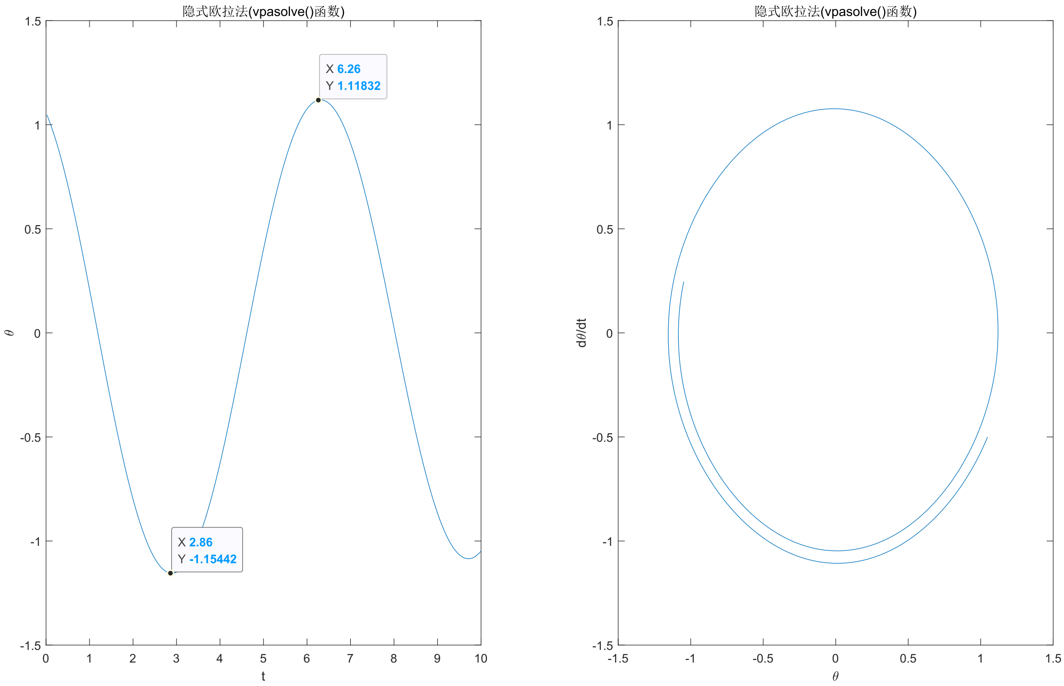
将初始迭代值代入隐式欧拉公式, 即：$\theta_{1(n+1)}^{(1)}=\theta_{1(n)}+h\theta_{2(n+1)}^{(0)}, \theta_{2(n+1)}^{(1)}=\theta_{2(n)}-h\sin[\theta_{1(n+1)}^{(0)}]$

$\therefore\ \theta_{1(n+1)}^{(k+1)}=\theta_{1(n)}+h\theta_{2(n+1)}^{(k)},\ \theta_{2(n+1)}^{(k+1)}=\theta_{2(n)}-h\sin[\theta_{1(n+1)}^{(k)}]$

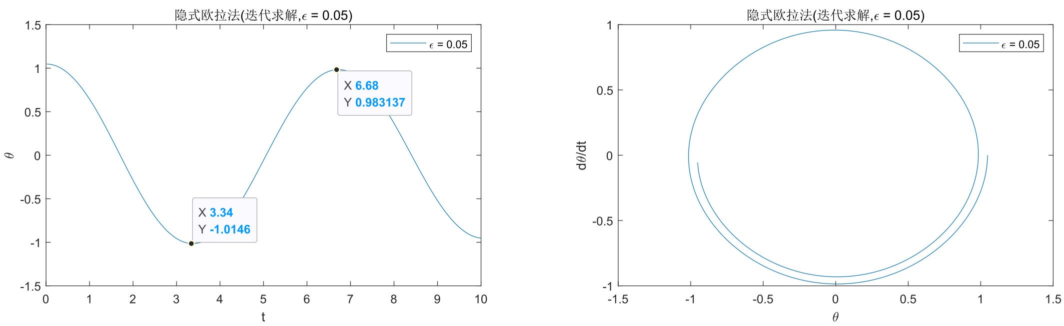
当$|\theta_{1(n+1)}^{(k+1)}-\theta_{1(n+1)}^{(k)}|<\epsilon,\ |\theta_{2(n+1)}^{(k+1)}-\theta_{2(n+1)}^{(k)}|<\epsilon$时迭代终止, 然后计算$\theta_{1(n+2)}$和$\theta_{2(n+2)}$

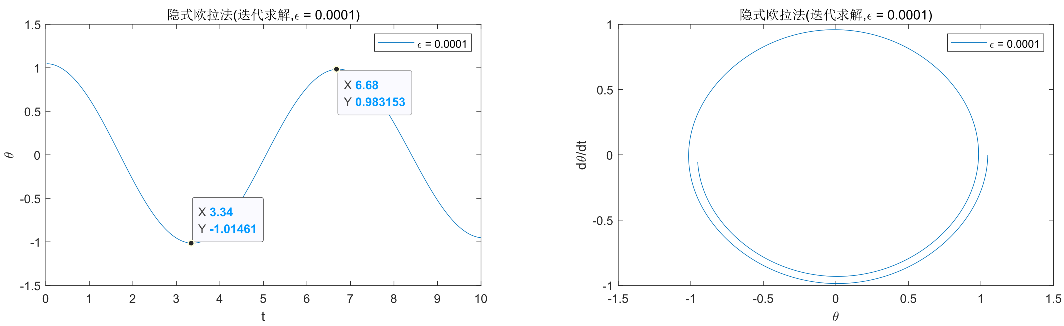
(2). 试验结果

- [方法一]



- [方法二]





(3). 结果分析

(i) [方法二$\epsilon = 0.05$]和[方法二$\epsilon = 0.0001$]的极值点和极值几乎相同

(ii) [方法一]与[方法二$\epsilon = 0.05$](或[方法二$\epsilon = 0.0001$])的极值点与极值有较明显的差别,其原因是[方法一]的vpasolve()函数的原理是直接求数值解,与[方法二]存在精度上的差异

(iii) 隐式欧拉法[方法一],[方法二]不保结构

(4). 代码

- [方法一]

```
clc;clf;
 syms x;
 h = 0.02;
 equ = (x + h^2*sin(x)==1);%equ = sin(x)==1;
 answ = vpasolve(equ)
h = 0.02;
n = 10/h;
syms x;
theta1 = zeros(1,n);
theta2 = theta1;
theta1(1) = pi/3;
theta2(1) = -1/2;
for i = 2:1:n
    equ = (x + h^2*sin(x) == theta1(i-1) + h*theta2(i-1));
    answ = vpasolve(equ);
    theta1(i) = answ;
    theta2(i) = theta2(i-1) - h*sin(answ);
end
subplot(1,2,2);
plot(theta1,theta2);
xlabel('\theta');
ylabel('d\theta/dt');
title('隐式欧拉法(vpasolve()函数)');
t = h:h:n*h;
subplot(1,2,1);
plot(t,theta1);
xlabel('t');
ylabel('\theta');
title('隐式欧拉法(vpasolve()函数)');
```

- [方法二] 以$\epsilon = 0.05$为例：

```
h = 0.02;
n = 10/h;
t = h:h:n*h;
theta1 = zeros(1,n);
theta2 = theta1;
epsilon = 0.05;
theta1(1) = pi/3;
for i = 2:1:n
    the1_0 = theta1(i-1) + h*theta2(i-1); % 初始迭代值
    the2_0 = theta2(i-1) - h*sin(theta1(i-1)); % 初始迭代值
    the1_ = theta1(i-1) + h*the2_0; % 第一次迭代值
    the2_ = theta2(i-1) - h*sin(the1_0); % 第一次迭代值
    while (abs(the_1_1 - the1_) > epsilon) && (abs(the_2_2 - the2_) > epsilon)
        the_1_1 = the1_; % 保留前一次迭代的值
        the_2_2 = the2_; % 保留前一次迭代的值
        the1_ = theta1(i-1) + h*the_2_2; % 第2,3,...次迭代值
        the2_ = theta2(i-1) - h*sin(the_1_1); % 第2,3,...次迭代值
    end
    theta1(i) = the1_;
    theta2(i) = the2_;
end
subplot(2,2,1);
plot(t,theta1);
xlabel('t');
ylabel('\theta');
legend('\epsilon = 0.05')
title('隐式欧拉法(迭代求解,\epsilon = 0.05)')
subplot(2,2,2);
plot(theta1,theta2);
xlabel('\theta');
ylabel('d\theta/dt');
legend('\epsilon = 0.05')
title('隐式欧拉法(迭代求解,\epsilon = 0.05)')
```

3. 梯形法

(1). 理论分析

$$\begin{pmatrix}\theta_{1(n+1)}\\ \theta_{2(n+1)}\end{pmatrix}=\begin{pmatrix}\theta_{1(n)}\\ \theta_{2(n)}\end{pmatrix}+\frac{h}{2}\begin{pmatrix}\theta'_{1(n)}+\theta'_{1(n+1)}\\ \theta'_{2(n)}+\theta'_{2(n+1)}\end{pmatrix}=\begin{pmatrix}\theta_{1(n)}\\ \theta_{2(n)}\end{pmatrix}+\frac{h}{2}\begin{pmatrix}\theta_{2(n)}+\theta_{2(n+1)}\\ -\sin[\theta_{1(n)}]-\sin[\theta_{1(n+1)}]\end{pmatrix} \qquad (**)$$

对一阶方程组$(*)$进行数值求解有多种求解方法,本试验共尝试两种方法：

[方法一]: $(**)$可化为 $\begin{cases}\theta_{1(n+1)}+\dfrac{h^2}{4}\sin[\theta_{1(n+1)}]=\theta_{1(n)}+h\theta_{2(n)}-\dfrac{h^2}{4}\sin[\theta_{1(n)}]\\ \theta_{2(n+1)}=\theta_{2(n)}-\dfrac{h}{2}\Big(\sin[\theta_{1(n)}]-\sin[\theta_{1(n+1)}]\Big)\end{cases}$

通过Matlab内置函数 vpasolve() 直接计算$\theta_{1(n+1)}$和$\theta_{2(n+1)}$的数值解

[方法二]: $(**)$可化为 $\begin{cases}\theta_{1(n+1)}=\theta_{1(n)}+\dfrac{h}{2}\theta_{2(n)}+\dfrac{h}{2}\theta_{2(n+1)}\\ \theta_{2(n+1)}=\theta_{2(n)}-\dfrac{h}{2}\Big(\sin[\theta_{1(n)}]+\sin[\theta_{1(n+1)}]\Big)\end{cases}$,迭代求解$\theta_{1(n+1)}$和$\theta_{2(n+1)}$：

先用显式欧拉法求初始迭代值$\theta^{(0)}_{1(n+1)}$和$\theta^{(0)}_{2(n+1)}$：即$\theta^{(0)}_{1(n+1)}=\theta_{1(n)}+h\theta_{2(n)}, \theta^{(0)}_{2(n+1)}=\theta_{2(n)}-h\sin[\theta_{1(n)}]$

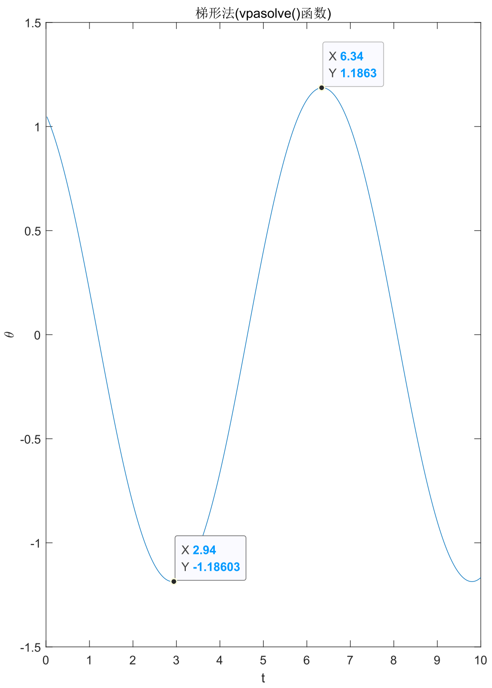
将初始迭代值代入梯形公式,即：$\theta^{(1)}_{1(n+1)}=\theta_{1(n)}+\dfrac{h}{2}\theta_{2(n)}+\dfrac{h}{2}\theta^{(0)}_{2(n+1)},\ \theta^{(1)}_{2(n+1)}=\theta_{2(n)}-\dfrac{h}{2}\Big(\sin[\theta_{1(n)}]+\sin[\theta^{(0)}_{1(n+1)}]\Big)$

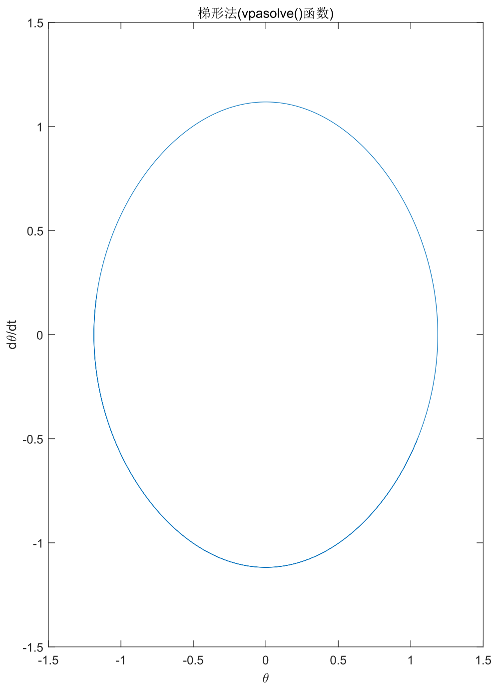
$\therefore \theta^{(k+1)}_{1(n+1)}=\theta_{1(n)}+\dfrac{h}{2}\theta_{2(n)}+\dfrac{h}{2}\theta^{(k)}_{2(n+1)},\ \theta^{(k+1)}_{2(n+1)}=\theta_{2(n)}-\dfrac{h}{2}\Big(\sin[\theta_{1(n)}]+\sin[\theta^{(k)}_{1(n+1)}]\Big)$

当$|\theta^{(k+1)}_{1(n+1)}-\theta^{(k)}_{1(n+1)}|<\epsilon,\ |\theta^{(k+1)}_{2(n+1)}-\theta^{(k)}_{2(n+1)}|<\epsilon$时迭代终止,然后计算$\theta_{1(n+2)}$和$\theta_{2(n+2)}$

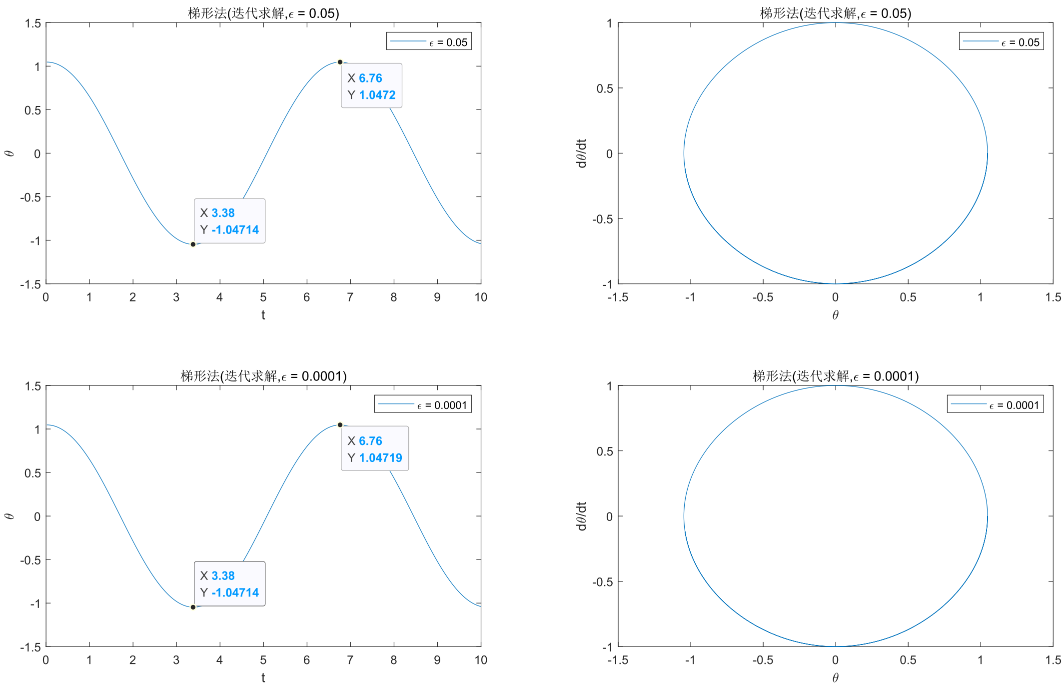
(2). 试验结果

- [方法一]





- [方法二]



(3). 结果分析

(i) [方法二$\epsilon = 0.05$]和[方法二$\epsilon = 0.0001$]的极值点和极值几乎相同

(ii) [方法一]与[方法二$\epsilon = 0.05$](或[方法二$\epsilon = 0.0001$])的极值点与极值有较明显的差别,其原因是[方法一]的vpasolve()函数的原理是直接求数值解,与[方法二]存在精度上的差异

(iii) 与隐式欧拉法相比,梯形法[方法二$\epsilon = 0.05$]和[方法二$\epsilon = 0.0001$]的极值的差异更小:

隐式欧拉法极值差异为:$t = 3.34$处相差$-1.0146 - (-1.01461) = 0.00001; t = 6.68$处相差$0.983137 - 0.983153 = -0.000015$

梯形法极值差异为:$t = 3.38$处相差$-1.04714 - (-1.04714) = 0; t = 6.76$处相差$1.0472 - 1.04719 = 0.00001$

推测梯形公式的迭代收敛速率大于隐式欧拉公式的迭代收敛速率

(iv) 梯形法[方法一],[方法二]保结构

(4). 代码

- [方法一]

```
clf;
h = 0.02;
n = 10/h;
syms x;
theta1 = zeros(1,n);
theta2 = theta1;
theta1(1) = pi/3;
theta2(1) = -1/2;
for i = 2:1:n
    equ = (x + h^2*sin(x)/4 == theta1(i-1) + h*theta2(i-1) - h^2*sin(theta1(i-1))/4);
    answ = vpasolve(equ);
    theta1(i) = answ;
    theta2(i) = theta2(i-1) - (h/2)*(sin(theta1(i-1)) + sin(answ));
end
subplot(1,2,2);
plot(theta1,theta2);
xlabel('\theta');
ylabel('d\theta/dt');
title('梯形法(vpasolve()函数)');
t = h:h:n*h;
subplot(1,2,1);
plot(t,theta1);
xlabel('t');
ylabel('\theta');
title('梯形法(vpasolve()函数)');
```

- [方法二] 以$\epsilon = 0.05$为例：

```matlab
h = 0.02;
n = 10/h;
t = h:h:n*h;
theta1 = zeros(1,n);
theta2 = theta1;
epsilon = 0.05;
theta1(1) = pi/3;
for i = 2:1:n
    the1_0 = theta1(i-1) + h*theta2(i-1); % 初始迭代值
    the2_0 = theta2(i-1) - h*sin(theta1(i-1)); % 初始迭代值
    the1_ = theta1(i-1) + (h/2)*theta2(i-1) + (h/2)*the2_0; % 第一次迭代值
    the2_ = theta2(i-1) - (h/2)*(sin(the1_0)+sin(theta1(i-1))); % 第一次迭代值
    while (abs(the_1_1 - the1_) > epsilon) && (abs(the_2_2 - the2_) > epsilon)
        the_1_1 = the1_; % 保留前一次迭代的值
        the_2_2 = the2_; % 保留前一次迭代的值
        the1_ = theta1(i-1) + (h/2)*theta2(i-1) + (h/2)*the_2_2; % 第2,3,...次迭代值
        the2_ = theta2(i-1) - (h/2)*(sin(the_1_1) + sin(theta1(i-1))); % 第2,3,...次迭代值
    end
    theta1(i) = the1_;
    theta2(i) = the2_;
end
subplot(2,2,1);
plot(t,theta1);
xlabel('t');
ylabel('\theta');
legend('\epsilon = 0.05')
title('梯形法(迭代求解,\epsilon = 0.05)')
subplot(2,2,2);
plot(theta1,theta2);
xlabel('\theta');
ylabel('d\theta/dt');
legend('\epsilon = 0.05')
title('梯形法(迭代求解,\epsilon = 0.05)')
```