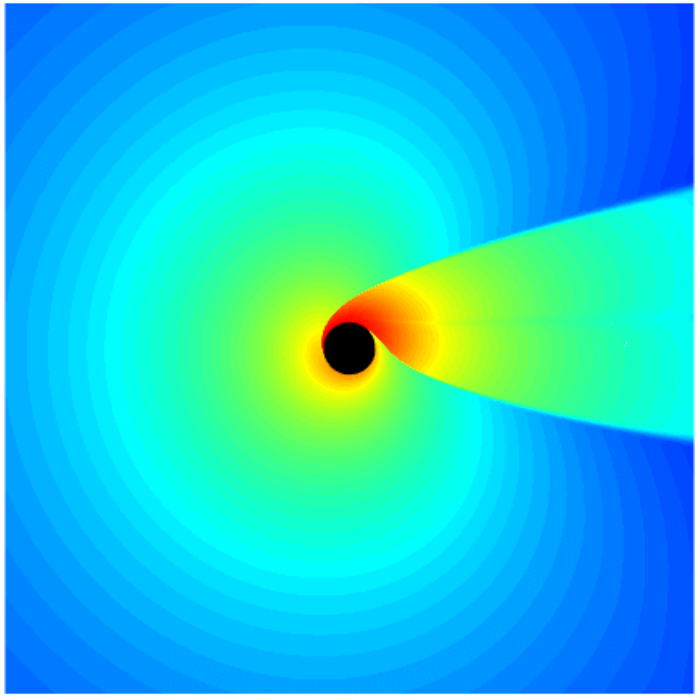
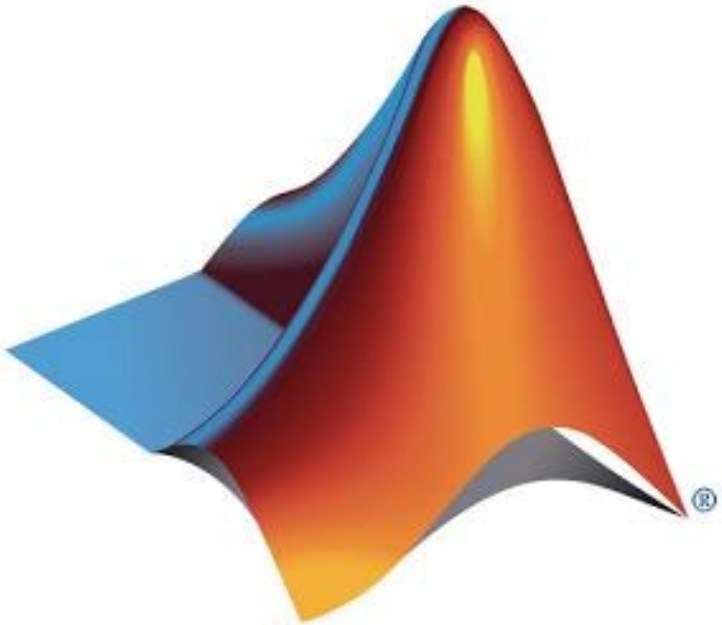
数学实验
Mathematical Experiments





实验十一：
微分方程实验
Differential Equation

什么是 微分方程?

◆实验的背景

一、问题背景

在许多实际问题中，需要考虑变化率问题，例如，在交通控制中需要考虑车流量的变化，在导弹飞行中需要研究导弹速率变化，在企业生产与销售时需要分析库存量的变化。在研究此类问题时，人们往往引入导数（或微分）来建立相关的数学模型，进行理论分析与数值求解。

◆实验的背景

含有未知函数及其某些阶的导数以及自变量本身的方程称为微分方程。微分方程是研究函数变化规律的有力工具，在科学研究、自动控制、经济管理、生态、环境、人口、交通、气象等领域中有广泛的应用。

微分方程中所含未知函数的导数的最高阶数（或微分最高阶数）称为微分方程的阶。若微分方程中的求导变量只有一个，称为常微分方程，否则称为偏微分方程。联系一些未知函数的一组微分方程称为微分方程组。

◆实验的背景

n阶常微分方程的一般形式为

$$F\left(x, y', y'', \ldots, y^{(n)}\right) = 0 \qquad (1)$$

若存在n阶可导函数y=y(x)，满足

$$F\left(x, y'(x), y''(x), \ldots, y^{(n)}(x)\right) = 0 \qquad (2)$$

则称函数y=y(x)为微分方程式(1)的解。若微分方程的解中所含任意常数的个数等于方程的阶数，则称其解为微分方程的通解。

◆实验的背景

当通解中各任意常数取定值时，所得到的解称为微分方程的特解。确定通解中任意常数的附加条件称为微分方程的初始条件（边值/边界条件），例如

$$y(x_0)=y_0, y'(x_0)=y_1,$$

$$y''(x_0)=y_2,\cdots,y^{(n-1)}(x_0)=y_{n-1} \quad (3)$$

◆实验的背景

若方程中未知函数及其各阶导数都是一次的，则称其为**线性常微分方程**，一般表示为

$$y^{(n)} + a_1(t)y^{(n-1)} + \cdots + a_{n-1}(t)y' + a_n(t)y = b(t) \quad (4)$$

若系数$a_i(t)(i = 1,2,\ldots,n)$均与t无关，则称其为**常系数**(或**定常、自治、时不变**)的.

◆实验的背景

线性常微分方程的解满足叠加性原理，从而它的求解可归结为求一个特解和相应齐次微分方程的解. 一阶变系数线性常微分方程总可用这一思路求得显式解. 高阶线性常系数微分方程可用特征根法求得相应齐次微分方程的基本解，再用常数变易法求特解.

◆实验的背景

高阶常微分方程初值问题可以化为一阶常微分方程组，已给一个n阶方程

$$y^{(n)} = f\left(t, y, y', \cdots, y^{(n-1)}\right), \qquad (5)$$

设$y_1 = y, y_2 = y', \cdots, y_n = y^{(n-1)}$，(5) 式化为一阶方程组

$$\text{方程组}\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ \cdots\cdots \\ y_{n-1}' = y_n \\ y_n' = f(t, y_1, y_2, \cdots, y_n) \end{cases}$$

为什么要研究数值解?

◆实验目的

对于许多实际问题，人们建立了相应的微分方程，并探讨了一些理论求解方法，并利用微分方程的解来解释实际现象，对特定问题进行分析、控制等。例如：

- 有些微分方程可直接通过积分求解。例如，一阶常系数线性常微分方程$y' = ay + b$。
- 有些常微分方程可用一些技巧(如分离变量法、积分因子法、常数变易法、降阶法等)化为可积分的方程而求得显式解（解析解）。

◆实验目的

但当前，理论成果主要体现在特定类型下的微分方程求解，多数情况下不能得到理论解（解析解）。

对于一些复杂微分方程，即使是一阶的，求解也很困难，有时候得到的是隐式解，有时候无法得到解析解。

当对于某些微分方程不能得到解析解时，人们一般去求微分方程的数值解。

◆实验目的

实验目的

(1)理解常微分方程数值求解原理，并利用计算机迭代求常微分方程数值解。

(2)熟悉MATLAB软件中常微分方程（组）的求解命令及用法。

(3)使用MATLAB解决一些常微分方程应用问题。

本实验仅限于探讨常微分方程（组），偏微分方程的数值解是更加复杂的课题。

什么是常微分方程数值解?

◆数值解简介

初值问题数值解

除常系数线性微分方程可用特征根法求解，少数特殊方程可用初等积分法求解外，大部分微分方程无显式解，应用中主要依靠数值解法，考虑一阶常微分方程组初值问题

$$y' = f(t, y),\ t_0 < t < t_f,$$

$$y(t_0) = y_0$$

其中$y = (y_1, y_2, \cdots, y_m)^T$,

$$f = (f_1, f_2, \cdots, f_m)^T,$$

$$y_0 = (y_{10}, y_{20}, \cdots, y_{m_0})^T$$

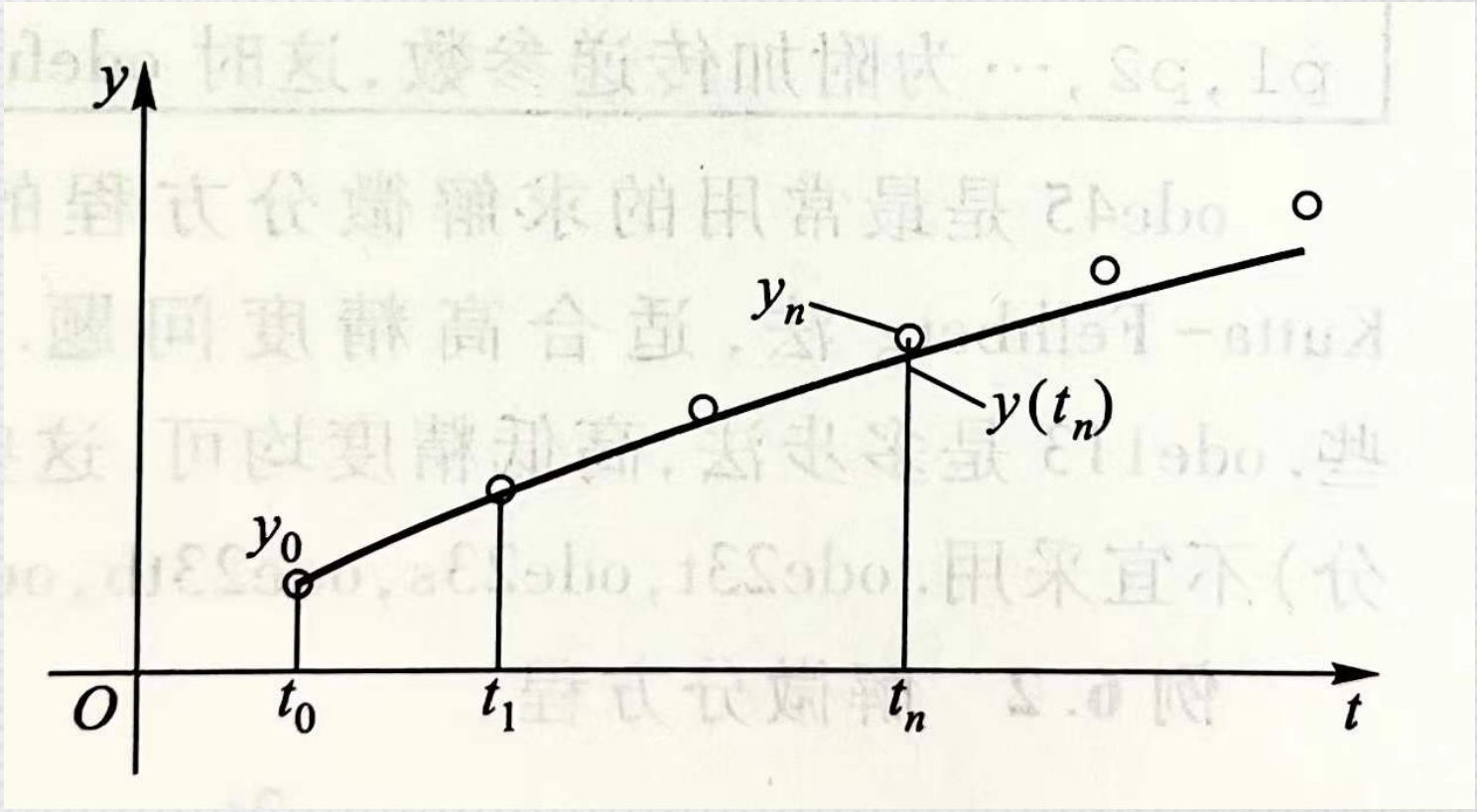
◆数值解简介

所谓**数值解**，就是寻求解$y(t)$在一系列离散结点$t_0 < t_1 < \cdots < t_n \leq t_f$上的近似值$y_k(k = 0,1,\ldots,n)$，

◆数值解简介

如图所示，称

$h_k = t_{k+1} - t_k$为步长，通常取为常量h.



常微分方程数值解

◆数值解简介

特别地取h_k为常数h,

则离散节点$t_n = t_0 + nh(n = 1,2,\cdots)$,

设在节点t_n处精确解$y(t_n)(n = 1,2,\ldots)$,

设计数值算法求$y(t_n)$的近似值$y_n(n = 1,2,\ldots)$.

◆数值解简介

使用数值方法求解初值问题需注意以下问题：

①数值方法的局部截断误差与相容性；②数值解y_n是否收敛于精确解$y(t_n)$以及估计整体截断误差$|y(x_n)-y_n|$。③数值方法的稳定性。

对以上问题的探讨，可参考相关专业书籍（数值分析、计算方法等）。

下面介绍几种经典的常微分方程数值求解方法，鉴于本课程的特点，算法叙述侧重算法原理与计算机实现，省略详细推导过程。

实验1：常微分方程数值方法简介

◆常微分方程数值方法简介

向前欧拉方法(Forward Euler Method),简称欧拉方法

欧拉方法的思路极其简单:

以等步长h为例,

在结点处用差商近似代替导数$y'(t_k) \approx \frac{y(t_{k+1})-y(t_k)}{h}$,

这样导出计算公式(称为Euler格式)

$$y_{k+1} = y_k + hf(t_k, y_k), \qquad k = 0,1,2,\cdots$$

式中:(t_0, y_0)为初始点。

也称为显式欧拉格式或向前欧拉格式。

◆常微分方程数值方法简介

向前欧拉方法(Forward Euler Method),简称欧拉方法

下列程序euler.m给出定步长Euler法解一阶常微分方程的计算程序，其使用格式为

[t,y]=euler(odefun,tspan,y0,h),

这里字符串odefun是用以表示$f(t,y)$的函数句柄或匿名函数,tspan $=[t_0,t_f]$表示自变量初值t_0和终值t_f,y0表示初值向量y_0，h是步长.输出列向量t表示结点$(t_0,t_1,\ldots,t_n)^T$,输出向量y表示数值解.

◆ 常微分方程数值方法简介

```
%M函数euler.m
function[t,y]=euler(odefun,tspan,y0,h)
t=tspan(1):h;tspan(2);y(1)=y0;
for  i=1:length(t)-1
y(i+1)=y(i)+h*odefun(t(i),y(i));
end
t=t';y=y';
end
```

◆常微分方程数值方法简介

向前欧拉方法(Forward Euler Method),简称欧拉方法

例如:

```
>>odefun=@(t,y)y-2*t/y;
>>[t,y]=euler(odefun,[0,4],1,0.01);
n=length(t);
e=sqrt(sum((sqrt(1+2*t)-y).^2)/n);
[n,e]
ans=
401.0000  0.3209
```

◆ 常微分方程数值方法简介

向后欧拉方法(Backward Euler Method),简称隐式欧拉方法(Implicit Euler Method)

$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}),\quad n = 0,1,2,\cdots$

- 向后欧拉方法通常稳定性比向前欧拉方法好
- 向后欧拉方法可能会涉及求解非线性方程（组）
- 欧拉方法只有一阶精度，所以实际应用效率比较差（需要用很密的网格）

◆ 常微分方程数值方法简介

实验 考虑微分方程：

$$\frac{dy}{dt} - \frac{2y}{t+1} = (t+1)^{\frac{5}{2}}, \qquad y(0) = 1$$

其解析解为：$y = (t+1)^2\left[\frac{2}{3}(t+1)^{\frac{3}{2}} + \frac{1}{3}\right]$

分别写出向前和向后欧拉方法的迭代格式

编程实现，观察误差收敛阶

◆常微分方程数值方法简介

- 欧拉方法只有一阶精度，所以实际应用效率比较差（需要用很密的网格）

- 常用高精度方法：
- Crank-Nicolson格式
- Runge-Kutta方法，多步法(Multi-Step Method)等

◆常微分方程数值方法简介

- Crank-Nicolson格式（梯形公式）

$$y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, y_{n+1})]$$

➢ 二阶精度：相比向前或向后欧拉公式算法精度要高，收敛速度快。

➢ 隐式的：很多情况下需解非线性代数方程（组）

- 改进欧拉公式

$$\begin{cases} \bar{y}_{n+1} = y_n + hf(t_n, y_n), \\ y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, \bar{y}_{n+1})], \end{cases} \quad n = 0,1,2,\cdots$$

◆常微分方程数值方法简介

实验 分别使用向前欧拉法与改进欧拉法求解微分方程：

$$\frac{dy}{dt} - \frac{2y}{t+1} = (t+1)^{\frac{5}{2}}, \qquad y(0) = 1$$

取步长h=0.1、h=0.01与h=0.001，给出数值解，并结合解析解，对求解结果进行数值对比。

◆ 常微分方程数值方法简介

改进欧拉法，即

$$\bar{y}_{n+1} = y_n + h\left[\frac{2y_n}{\mathrm{t}_n + 1} + (\mathrm{t}_n + 1)^{\frac{5}{2}}\right]$$

$$z_1 = \frac{2y_n}{\mathrm{t}_n + 1} + (\mathrm{t}_n + 1)^{\frac{5}{2}}$$

$$z_2 = \frac{2\bar{y}_{n+1}}{\mathrm{t}_{n+1} + 1} + (\mathrm{t}_{n+1} + 1)^{\frac{5}{2}}$$

$$y_{n+1} = y_n + \frac{h}{2}[z_1 + z_2], \qquad n = 0,1,2,\ldots$$

◆常微分方程数值方法简介

- 龙格-库塔（Runge-Kutta）方法

对梯形公式实际是采用t_n与t_{n+1}两点处的斜率取算术平均代替曲线斜率f(t, y)。更进一步，对区间$[t_n, t_{n+1}]$内任一点$t_{n+p} = t_n + ph(0 < p \leq 1)$，用$t_n$与$t_{n+1}$两个点的斜率值$K_1$、$K_2$加权平均得到平均斜率$K^*$，即

$$K^* = (1-\lambda)K_1 + \lambda K_2$$

式中：λ为待求参数。

◆常微分方程数值方法简介

- 龙格-库塔（Runge-Kutta）方法

可以得到以下选代格式：

$$\begin{cases} y_{n+1} = y_n + h[(1-\lambda)K_1 + \lambda K_2] \\ K_1 = f(x_n, y_n) \\ K_2 = f(x_n + ph, y_n + phK_1) \end{cases}$$

在数值计算中，适当选取参数λ、p，可使上式有较高精度。通过泰勒展开，若具有二阶精度，满足$\lambda p = \frac{1}{2}$。得到二阶龙格-库塔格式。

◆常微分方程数值方法简介

- 龙格-库塔（Runge-Kutta）方法

类似地，可涉及更高阶精度的龙格-库塔格式。例如，经典的四阶龙格-库塔格式：

$$\begin{cases} y_{n+1} = y_n + \dfrac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 = f(t_n, y_n) \\ K_2 = f\left(t_n + \dfrac{h}{2}, y_n + \dfrac{h}{2}K_1\right) \\ K_3 = f\left(t_n + \dfrac{h}{2}, y_n + \dfrac{h}{2}K_2\right) \\ K_4 = f(t_n + h, y_n + hK_3) \end{cases}$$

◆ 常微分方程数值方法简介

实验： 用经典的四阶龙格-库塔格式求解微分方程：

$$\frac{dy}{dt} - \frac{2y}{t+1} = (t+1)^{\frac{5}{2}}, \qquad y(0) = 1$$

其解析解为：$y = (t+1)^2 \left[\frac{2}{3}(t+1)^{\frac{3}{2}} + \frac{1}{3}\right]$

◆ 常微分方程数值方法简介

实验： 分别使用四阶龙格一库塔法与改进欧拉法求解微分方程：

$$y' = y - \frac{2x}{y}, y(0) = 1$$

取步长h=0.1与h=0.01，给出两种方法的数值解，并结合解析解，对求解结果进行数值比较。

◆常微分方程数值方法简介

解： 该问题解析解为$y=\sqrt{1+2t}$，数值求解程序如下：

```
function[y1,y2,y3]=Exp11_1c(a,b,k)
%a起始点，b选代终止点，h=(b-a)/k
h=(b-a)/k;x=a:h:b:[m,n]=size(x);
y1=zeros(m,n);y2=zeros(m,n):y3=zeros(m,n);
y1(1)=1:y2(1)=1:y3(1)=1;
for  i=1:n-1   %改进欧拉法
     x1=x(i);x2=x(i+1);
     z1=y1(i)-2*x1/y1(i);
     yy=y1(3)+h*z1;
     z2=yy-2*x2/yy;
     y1(i+1)=y1(i)+h*0.5*(z1+z2);
end
```

◆常微分方程数值方法简介

```
for  i=1:n-1   %四阶龙格一库塔法
     x1=x(i);x2=x(i+1);xx=x1+h/2;
     k1=y2(i)-2*x1/y2(i);
     yy=y2(i)+h*k1/2;
     k2=yy-2*xx/yy;
     yy=y2(i)+h*k2/2;
     k3=yy-2*xx/yy;
     yy=y2(i)-h*k3;
     k4=yy-2*x2/yy;
     y2(i+1)=y2(1)+h/6*(k1+2*k2+2*k3+k4);
end
for i=1:n    %精确解
y3(i)=sgrt(1+2*x(i));
end
```

◆ 常微分方程数值方法简介

从表中数据可以看出，四阶龙格-库塔法精度较高。

t_n	h=0.1		h=0.01		精确解
	改进欧拉法	四阶龙格-库塔法	改进欧拉法	四阶龙格-库塔法	
0	1.0000000000	1.0000000000	1.0000000000	1.0000000000	1.0000000000
0.1	1.0959090909	1.0954455317	1.0954497407	1.0954451150	1.0954451150
0.2	1.1840965692	1.1832167455	1.1832247992	1.1832159567	1.1832159566
0.3	1.2662013609	1.2649122283	1.2649240883	1.2649110642	1.2649110641
0.4	1.3433601515	1.3416423538	1.3416582109	1.3416407866	1.3416407865
0.5	1.4164019285	1.4142155779	1.4142358094	1.4142135626	1.4142135624
0.6	1.4859556024	1.4832422228	1.4832673781	1.4832396977	1.4832396974
0.7	1.5525140913	1.5491964523	1.5492272567	1.5491933388	1.5491933385
0.8	1.6164747828	1.6124553497	1.6124927197	1.6124515500	1.6124515497
0.9	1.6781663637	1.6733246590	1.6733697285	1.6733200535	1.6733200531
1.0	1.7378674010	1.7320563652	1.7321105201	1.7320508081	1.7320508076

实验2：Matlab求解常微分方程的内置函数/指令

◆Matlab内置函数/指令

- 解析解指令：符号计算dsolve
- 数值解指令：见下表

名称	特点	说明
ode45	单步法;4-5阶龙格-库塔方法;精度中等	大部分场合的首选算法
ode23	单步法;2-3阶龙格-库塔方法;低精度	适用于精度较低情形
ode113	多步法;Adams算法;高低精度	计算时间比ode45短
ode23t	梯形方法	适度刚性情形
ode15s	多步法;Gear's反向数值积分;精度中等	若ode45失效时，可尝试使用
ode23s	单步法;二阶Rosebrock 算法;低精度	当精度较低时，计算时间比ode15s短
ode23tb	梯形算法;低精度	当精度较低时，计算时间比ode15s短
ode15i	可变秩方法;低精度	完全隐式微分方程求解

◆MATLAB求解微分方程

一、解析解

MATLAB软件中求解微分方程解析解的函数为dsolve，调用格式：

dsolve('eqn1','eqn2',…,'con1','con2',…,'v')：

eqn1、eqn2、…为输入方程(组);con1、con2、…为初始条件;v表示求导变量,省略时系统默认变量为t。在输入方程或初始条件时,用Dy表示y关于自变量的一阶导数,用D2y表示y关于自变量的二阶导数,用Dny表示y关于自变量的n阶导数。

◆MATLAB求解微分方程

1. 通解

例 求下列微分方程的通解:

(1) $y' = e^{2t} - \sin 3t$; (2) $\frac{dy}{dx} = x^2 + y$。

```
>>y1=dsolve('Dy =exp(2*t)-sin(3*t)')

y1

=1/2*exp(2*t)+1/3*cos(3*t)+C1

>>y2=dsolve('Dy=x2+y','x')

y2=

-2-2*x-x2+exp(x)*C1
```

◆MATLAB求解微分方程

上述求解结果表明：

(1) 通解$y=\frac{1}{2}e^{2t}+\frac{1}{3}\cos 3t+C_1$；

(2) 通解$y=-2-2x-x^2+C_1e^x$.

在求解过程中，要指明自变量，若省略表示自变量为t，例如(2)，执行：

```
>>y2=dsolve('Dy=x2+y')
y2=
-x2+exp(t)*C1
```

◆MATLAB求解微分方程

2. 特解

例 求微分方程$y' = -y + 2x + 1, y(0) = 1$

的特解。

```
>>y=dsolve('Dy=-y+2*x+1','y(0)=1','x')
y=
-1+2*x+2*exp(-x)
```

即表示微分方程特解为

$$y = -1 + 2x + 2e^{-x}$$

◆MATLAB求解微分方程

3. 高阶微分方程

例 求微分方程$y'' - 5y' + 6y = xe^{2x}$的通解。

```
>>y=dsolve('D2y-5 * Dy+6 *y=x*exp(2*x)','x')

y=

exp(2*x)*C2+exp(3*x)*C1-x*exp(2*x)-

1/2*x^2*exp(2*x)
```

即表示微分方程通解为$y = c_2e^{2^x} + c_1e^{3x} - \frac{1}{2}x^2e^{2x}$

◆MATLAB求解微分方程

4. 微分方程组

例 求微分方程组$\begin{cases}\frac{dy}{dx}=x+2y-3z\\ \frac{dz}{dx}=3x+y-2z\end{cases}$的通解。

```
[Y,Z]=dsolve('Dy=x+2*y-3 *z','Dz=3 *x+y-
2*z','x')
```

```
Y=exp(x)*C2+exp(-x)*C1-1+7*x
```

```
Z=1/3*exp(x)*C2+exp(-x)*C1-3+5*x
```

◆MATLAB求解微分方程

二、数值解

1. 数值求解函数使用说明

当难以求得微分方程的解析解时，可以求其数值解。在求微分方程数值解方面，MATLAB具有多个函数，将其统称为solver，其一般格式为：

[T, Y]=solver(odefun, tspan, y0)：表示求解微分方程$y' = f(t, y)$的数值解，odefun为微分方程中的f(t, y)项；tspan为求解区间，要获得问题在指定点$t_0, t_1, \cdots, t_f$的解

◆MATLAB求解微分方程

可取 tspan = $[t_0, t_1, \cdots, t_f]$ （分量单调）；y0为初始条件。solver为数值求解微分方程函数，下表给出了具体名称与用法说明。

名称	特点	说明
ode45	单步法;4-5阶龙格-库塔方法;精度中等	大部分场合的首选算法
ode23	单步法;2-3阶龙格-库塔方法;低精度	适用于精度较低情形
ode113	多步法;Adams算法;高低精度	计算时间比ode45短
ode23t	梯形方法	适度刚性情形
ode15s	多步法;Gear's反向数值积分;精度中等	若ode45失效时，可尝试使用
ode23s	单步法;二阶Rosebrock 算法;低精度	当精度较低时，计算时间比ode15s短
ode23tb	梯形算法;低精度	当精度较低时，计算时间比ode15s短
ode15i	可变秩方法;低精度	完全隐式微分方程求解

◆MATLAB求解微分方程

在求解过程中有时需要对求解算法和控制条件进行进一步设置,这时可以调用带op-tions参数的函数适用格式:

[T,Y]=solver(odefun,tspan,y0,options):options用于对求解算法与控制条件进行设置。初始options变量可以通过odeset获取,调用格式:

options=odeset('name1',value1,'name2',value2,…)

常用控制参数主要有'AbsTol'(绝对误差限1e-6)、'RelTol'(相对误差限1e-3)、'MaxStep'(求解方程最大允许的步长)、'Mass'(微分方程中的质量矩阵,可用于描述微分方程)。

◆MATLAB求解微分方程

2. 求解算例

例1 求微分方程初值问题 $\begin{cases} \frac{dy}{dx} = -2x + 2x^2 + 2x \\ y(0) = 1 \end{cases}$ 的数值解，求解范围为[0,0.5]。

编制函数文件：

```
function f=odefun(x,y)
f=-2*y+2*x^2+2*x;
```

◆4.3 MATLAB求解微分方程

命令窗口下执行:

```
>>[x,y]=ode23('odefun',[0,0.5],1);
>>[x';y']
ans=
0  0.0400  0.0900  0.1400  0.1900  0.3900  0.2400
0.2900  0.3400  0.4400  0.4900  0.5000
1.0000  0.9247  0.8434  0.7754  0.7199  0.6105  0.6764
0.6440  0.6222  0.6084  0.6154  0.6179
>>plot(x,y,'o -')
```

◆MATLAB求解微分方程

解曲线如图所示。

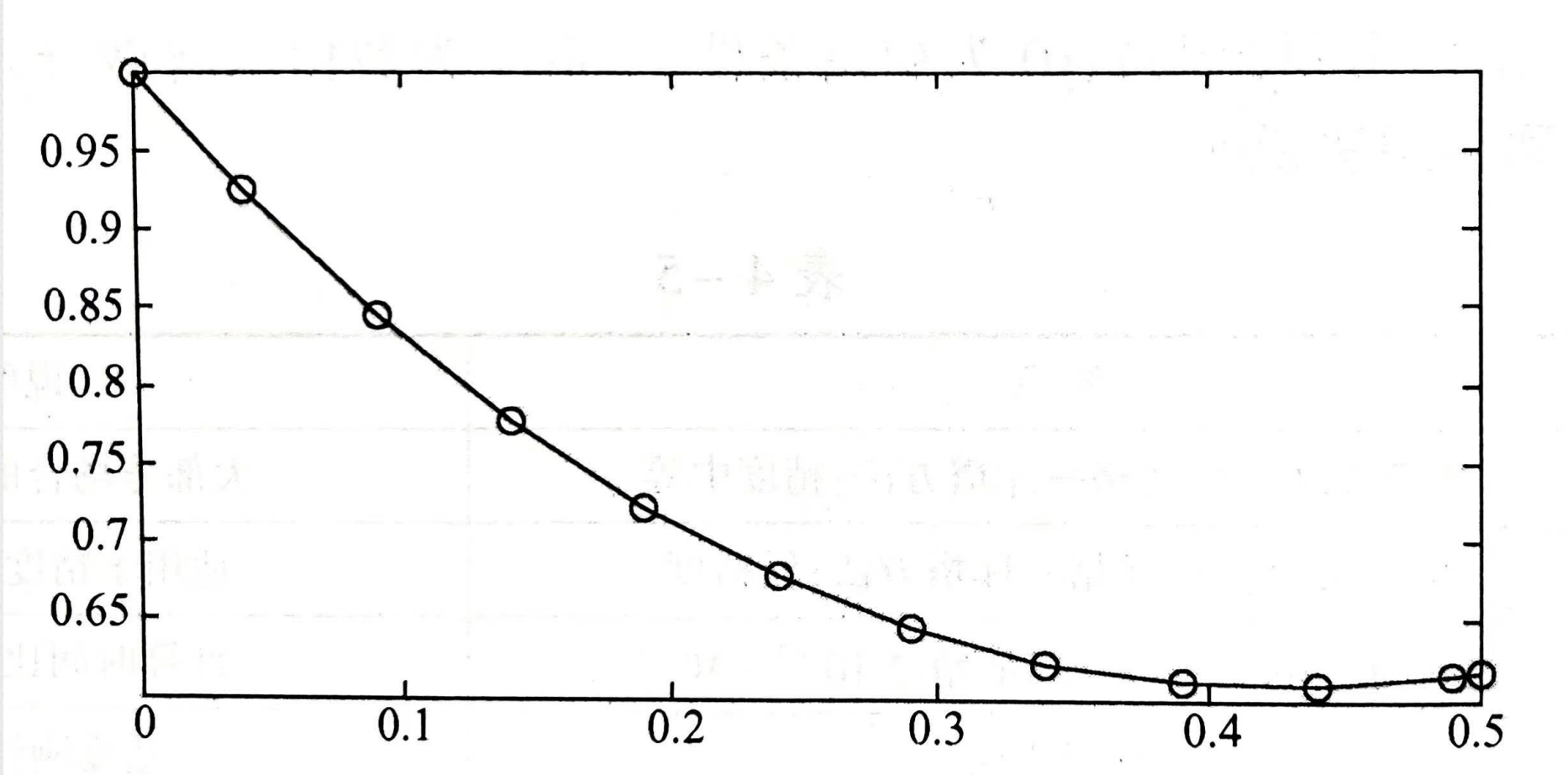


图4-1 例4-8数值解曲线

◆MATLAB求解微分方程

例2 求解

$$\begin{cases} y_1' = -0.01y_1 - 99.99y_2, y_1(0) = 2, \\ y_2' = -100y_2, y_2(0) = 1 \end{cases} \quad (0 < t < 400).$$

解 先写M函数odefun.m.

```
%M函数odefun.m
function f=odefun(t,y)
    f(1)=-0.01*y(1)-99.99*y(2);
    f(2)=-100*y(2);
    f=f(:);    %这样可保证f输出时为列向量
end
```

◆MATLAB求解微分方程

例2 求解

$$\begin{cases} y_1' = -0.01y_1 - 99.99y_2, y_1(0) = 2, \\ y_2' = -100y_2, y_2(0) = 1 \end{cases} \quad (0 < t < 400).$$

若用ode45求解，则

```
>>clear;tic;
[t,y]=ode45(@odefun,[0,400],[2,1]);
toc,plot(t,y);
```

历时0.875150s %计算时间会因计算机软硬件情况不同有区别可见计算速度太慢.

◆MATLAB求解微分方程

例2 求解

$$\begin{cases} y_1' = -0.01y_1 - 99.99y_2, y_1(0) = 2, \\ y_2' = -100y_2, y_2(0) = 1 \end{cases} \quad (0 < t < 400).$$

这个问题的特点是y2下降很快，而y1下降太慢. 一方面，由于y2下降太快，为了保证数值稳定性，步长h需足够小；另一方面，由于y1下降太慢，为了反映解的完整性，时间区间需足够长，这就造成计算量太大. 这类方程组称为刚性方程组或病态方程组. ode45不适用于病态方程组，下面我们用ode15s求解.

◆MATLAB求解微分方程

例2 求解

$$\begin{cases} y_1' = -0.01y_1 - 99.99y_2, y_1(0) = 2, \\ y_2' = -100y_2, y_2(0) = 1 \end{cases} \quad (0 < t < 400).$$

```
>>clear;tic;
[t,y]=ode15s(@eg6_6fun,[0,400],[2,1]);
toc,plot(t,y);
```

计算速度提高。

◆MATLAB求解微分方程

例3 求解描述震荡器的经典Ver der Pol方程：

$$\frac{d^2y}{dt^2}-\mu(1-y^2)\frac{dy}{dt}+y=0, y(0)=1, y'(0)=0$$

解：这是一个二阶非线性方程，用Matlab数值求解指令不能直接求解。

◆MATLAB求解微分方程

例3 求解描述震荡器的经典Ver der Pol方程：

$$\frac{d^2y}{dt^2}-\mu(1-y^2)\frac{dy}{dt}+y=0, y(0)=1, y'(0)=0$$

解：这是一个二阶非线性方程，用Matlab数值求解指令不能直接求解。但可以通过下面的变换将二阶方程化为一阶方程组，则可求解。令$x_1=y, x_2=\frac{dy}{dt}$，得

$$\begin{cases}\dfrac{dx_1}{dt}=x_2, x_1(0)=1\\ \dfrac{dx_2}{dt}=\mu(1-{x_1}^2)x_2-x_1, x_2(0)=0\end{cases}$$

◆4.3 MATLAB求解微分方程

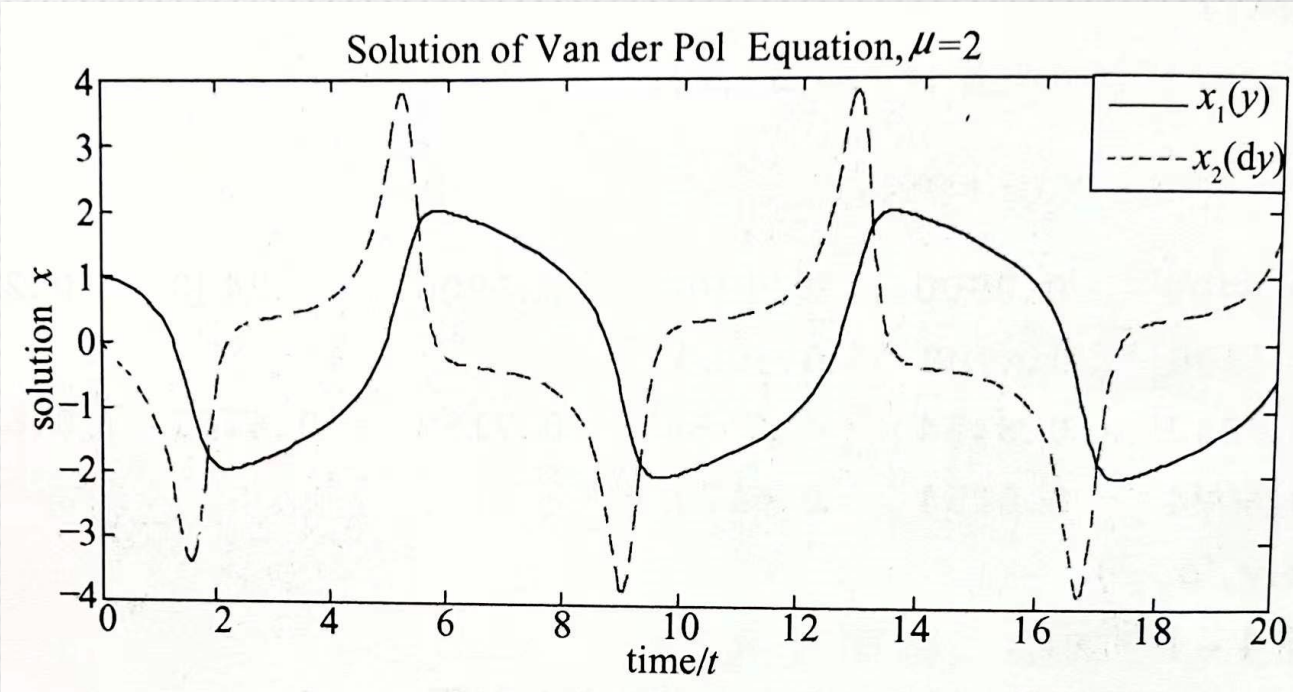
编制函数文件(μ=2):

```
ffunction f=odefun(t,x)
mu=2;
f=[x(2);mu*(1-x(1)^2)*x(2)-x(1)];
```

◆4.3 MATLAB求解微分方程

```
[t,x]=ode45(@odefun,[0,20],[1;0]);
plot(t,x(:,1),'-',t,x(:,2),'--');
title('Solution of Van der Pol Equation,/mu =2');
xlabel('time t');ylabel('solution x');
legend('x_1(y)','x_2(dy)');
```

解曲线如图所示。



◆MATLAB求解微分方程

例4 已知Lorenz状态方程为

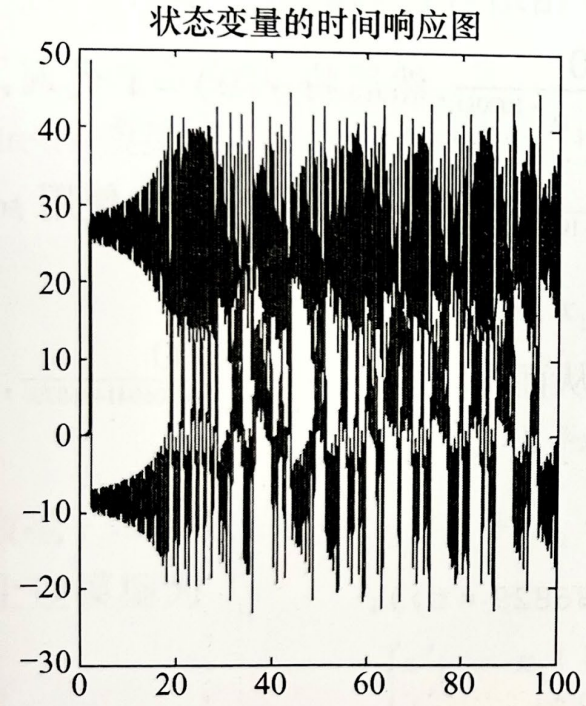
$$\begin{cases} \dot{x}_1(t) = -\beta x_1(t) + x_2(t)x_3(t) \\ \dot{x}_2(t) = -\rho x_2(t) + \rho x_3(t) \\ \dot{x}_3(t) = -x_1(t)x_2(t) + \sigma x_2(t) - x_3(t) \end{cases}$$

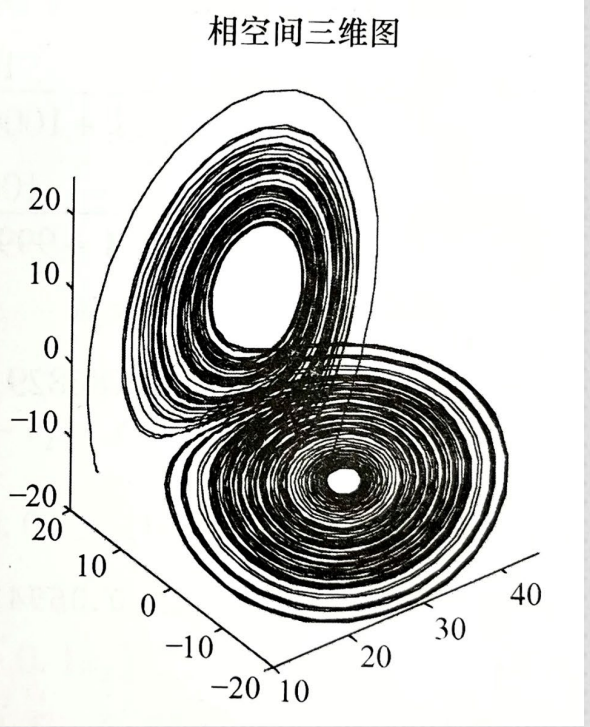
初始值为

$$\begin{cases} x_1(0) = 0 \\ x_2(0) = 0 \\ x_3(0) = 10^{-10} \end{cases}$$

$\beta = \frac{8}{3}, \rho = 10, \sigma = 28$, 求解该微分方程组。

◆MATLAB求解微分方程





编制函数文件：

```
function xx=odefun(t,x)
beta=8/3;rou=10;sig=28;
xx=[-beta*x(1)+x(2)*x(3);-rou*x(2)+rou*x(3);
-x(1)*x(2)+sig*x(2)-x(3)];
```

◆MATLAB求解微分方程

```
x0=[0;0;1e-10];
[t,x]=ode45(@odefun,[0,100],x0);
subplot(1,2,1);
plot(t,x);
title('状态变量的时间响应图');
subplot(1,2,2);
plot(x(:,1),x(:,2),x(:,3));
>>plot3(x(:,1),x(:,2),x(:,3));
axis([10,45,-20,20,-20,25]);
title('相空间三维图');
```

◆MATLAB求解微分方程

例5 解微分方程组

$$\begin{cases} x' = -x^3 - y, x(0) = 1, \\ y' = x - y^3, y(0) = 0.5, \end{cases} \quad 0 < t < 30. \qquad (6.9)$$

解 将变量x,y合写成向量变量x,先写M函数eg6_3fun.m:

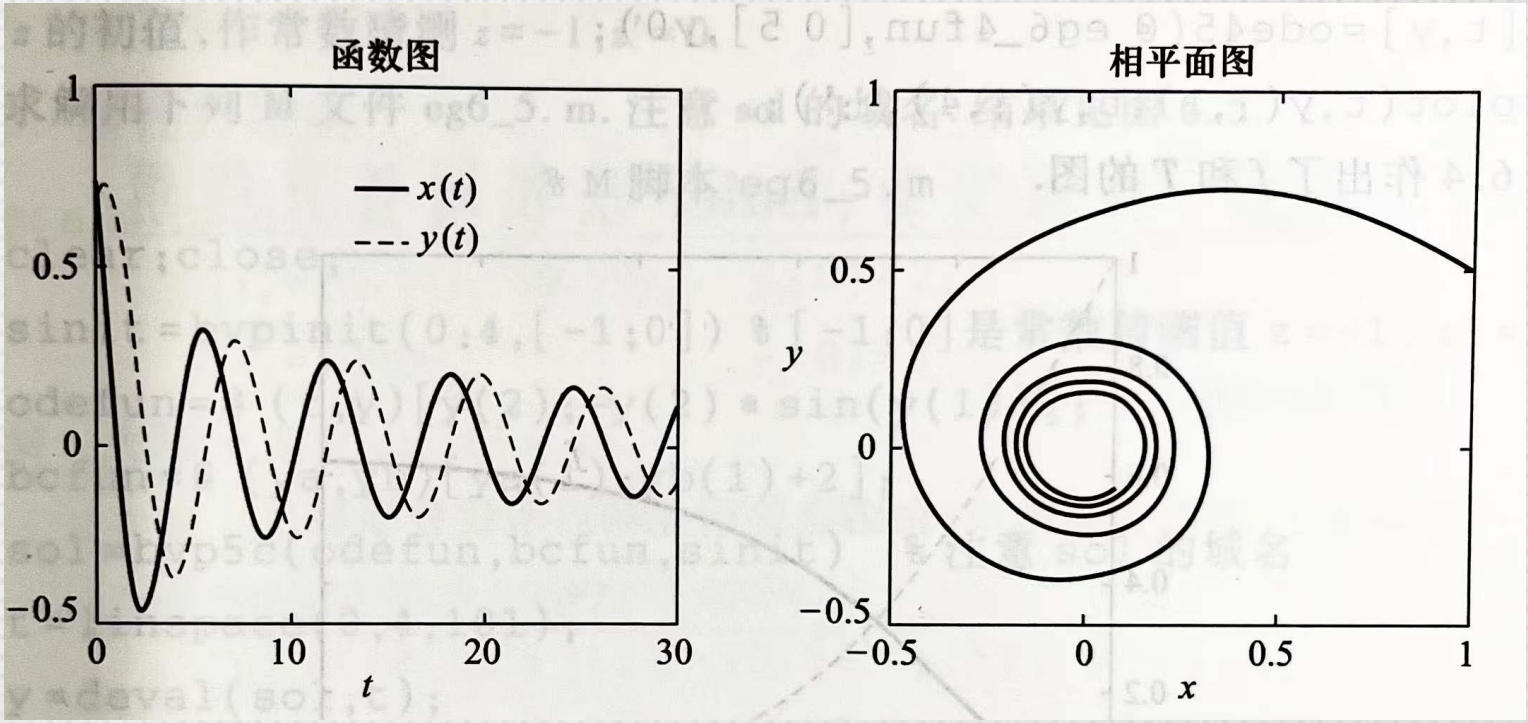
```
%M函数eg6_3fun.m
function f=eg6_3fun(t,x)
f(1)=-x(1)^3-x(2);
f(2)=x(1)-x(2)^3;
f=f(:);      %保证f为列向量
end
```

MATLAB求解微分方程

再执行

```
clear;[t,x]=ode45(@eg6_3fun,[0 30],[1;0.5]);
subplot(1,2,1);plot(t,x(:,1),t,x(:,2),':');
subplot(1,2,2);plot(x(:,1),x(:,2));
```

作出了解函数图和相平面图.



◆MATLAB求解微分方程

例6 求解微分方程组(竖直加热板的自然对流)

$$\begin{cases} \dfrac{d^3f}{d\eta^3}+3f\dfrac{d^2f}{d\eta^2}-2\left(\dfrac{df}{d\eta}\right)^2+T=0 \\ \dfrac{d^2T}{d\eta^2}+2.1f\dfrac{dT}{d\eta}=0 \end{cases}$$

已知当$\eta=0$时，$f=0,\frac{df}{d\eta}=0,\frac{d^2f}{d\eta^2}=0.68,T=1,\frac{dT}{d\eta}=-0.5.$

◆MATLAB求解微分方程

解 首先引入辅助变量

$$t=\eta, y_1=f, y_2=\frac{df}{d\eta}, y_3=\frac{d^2f}{d\eta^2}, y_4=T, y_5=\frac{dT}{d\eta},$$

化为一阶方程组

$$\begin{cases}\frac{dy_1}{dt}=y_2,\\ \frac{dy_2}{dt}=y_3,\\ \frac{dy_3}{dt}=-3y_1y_3+2y_2^2-y_4,\\ \frac{dy_4}{dt}=y_5,\\ \frac{dy_5}{dt}=y_5-2.1y_1y_5,\end{cases}$$

MATLAB求解微分方程

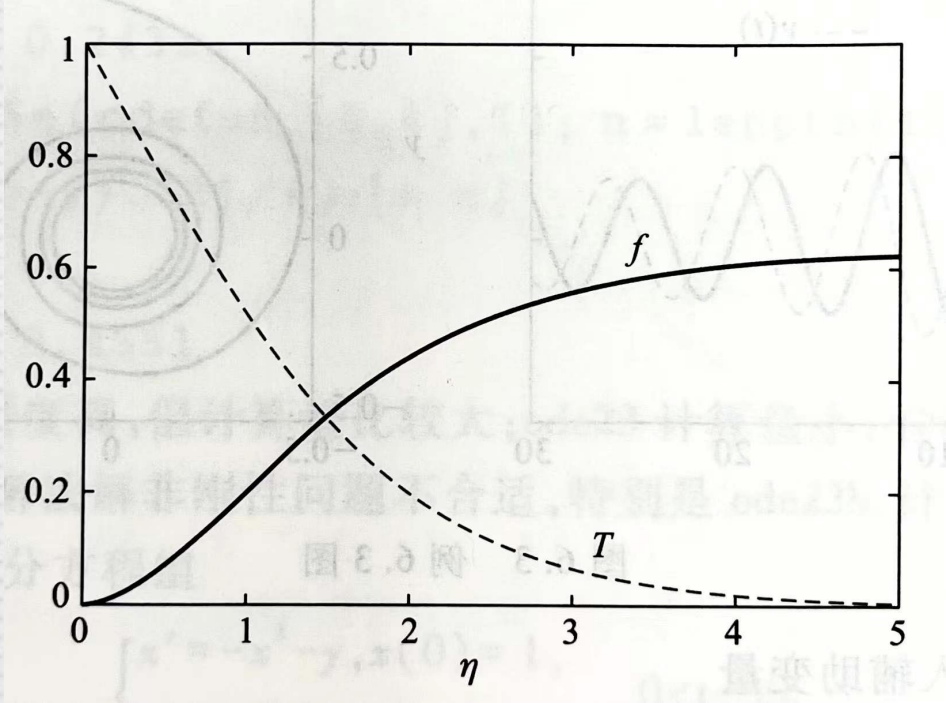
先写M函数eg6_4.m.

```
function f=eg6_4fun(t,y)
f=[y(2);y(3);-3*y(1)*y(3)+2*Y(2)^2-
y(4);y(5);-2.1*y(1)*y(5)];%注意要保证f为列
向量
end
```

MATLAB求解微分方程

再执行

```
y0=[0,0,0.68,1,-0.5];
[t,y]=ode45(@eg6_4fun,[0 5],y0);
plot(t,y(:,1),t,y(:,4),':');
```



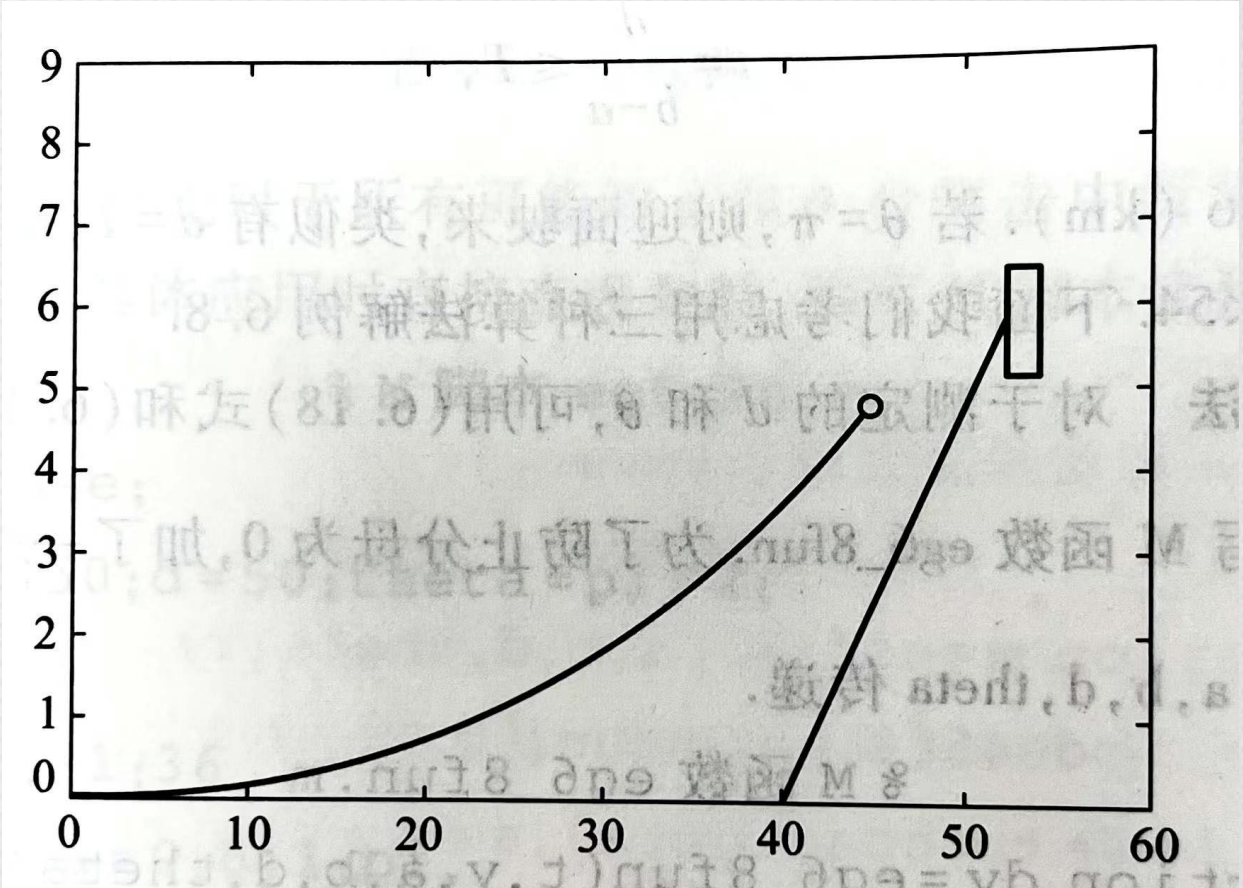
实验3　应用性实验：导弹系统改进问题

◆ 建模实验:导弹系统的改进

海军方面要求改进现有的舰对舰导弹系统. 目前的电子系统能迅速测出敌舰的种类、位置以及敌舰行驶速度和方向，且导弹自动制导系统能保证在发射后任一时刻都能对准目标. 根据情报，这种敌舰能在我军舰发射导弹后T_{min}作出反应并摧毁导弹. 现在要求改进电子导弹系统使其能自动计算出敌舰是否在有效打击范围之内.

◆建模实验:导弹系统的改进

设军舰发射导弹时位置在坐标原点(如图),敌舰在x轴正向d km处,其行驶速度为a km/h,方向与x轴夹角为θ,导弹水平飞行线速度为b km/h.问题的关键是求出导弹击中敌舰的时间.



◆建模实验:导弹系统的改进

设t时刻时导弹位置为(x(t),y(t)),那么线速度

$$\sqrt{\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2}=b$$

易知t时刻敌舰位置为$(d+at\cos\theta, at\sin\theta)$,为了保持对准目标,导弹轨迹切线方向应为

$$\frac{dy}{dx}=\frac{at\sin\theta-y(t)}{d+at\cos\theta-x(t)}$$

◆ 建模实验:导弹系统的改进

得下列微分方程组:

$$\frac{dx}{dt}=\frac{b}{\sqrt{1+\left(\frac{dy}{dx}\right)^2}}=\frac{b}{\sqrt{1+\left(\frac{at\sin\theta-y(t)}{d+at\cos\theta-x(t)}\right)^2}},$$

$$\frac{dy}{dt}=\frac{b}{\sqrt{1+\left(\frac{dx}{dy}\right)^2}}=\frac{b}{\sqrt{1+\left(\frac{d+at\cos\theta-x(t)}{at\sin\theta-y(t)}\right)^2}},$$

初始条件x(0)=0, y(0)=0. 对于给定的a, b, d, θ进行计算.

击中目标的条件是什么?

◆ 建模实验:导弹系统的改进

得下列微分方程组:

$$\frac{dx}{dt}=\frac{b}{\sqrt{1+\left(\frac{dy}{dx}\right)^2}}=\frac{b}{\sqrt{1+\left(\frac{at\sin\theta-y(t)}{d+at\cos\theta-x(t)}\right)^2}}, \tag{6}$$

$$\frac{dy}{dt}=\frac{b}{\sqrt{1+\left(\frac{dx}{dy}\right)^2}}=\frac{b}{\sqrt{1+\left(\frac{d+at\cos\theta-x(t)}{at\sin\theta-y(t)}\right)^2}}, \tag{7}$$

初始条件x(0)=0, y(0)=0. 对于给定的a, b, d, θ进行计算. 若x(t)满足$x(t)\geq d+at\cos\theta$,

则认为已击中目标. 如果$t\leq T$, 那么敌舰在打击范围内, 可以发射.

◆建模实验:导弹系统的改进

问题: 在导弹系统中设a=90km/h, b=450km/h, T=0. 1h. 现要求d,θ的有效范围.

解 有两个极端情形容易算出. 若$\theta = 0$，则敌舰正好背向行驶，即沿x轴正向行驶，此时导弹直线飞行，击中时间

$$t = \frac{d}{b-a} \leq T$$

得d=T(b−a)=36(km). 若$\theta = \pi$，则迎面驶来，类似有d=T(a+b)=54(km).

◆建模实验:导弹系统的改进

一般地，应有36<d<54. 下面我们考虑用三种算法

(1)在线算法 对于测定的d和0，可用(6)式和(7)式计算出t.

比如$d=50,\theta=\frac{\pi}{2}$, 写M函数eg6_8fun. 为了防止分母为0, 加了一个小正数1e-8, 并且使用附加参数a, b, d, theta传递.

```
function dy=eg6_8fun(t,y,a,b,d,theta)
    dydx=(a*t*sin(theta)-y(2)+1e-8)/...
    (abs(d+a *t*cos(theta)-y(1))+1e-8);
    dy(1)=b/(1+dydx^2)^0.5;
    dy(2)=b/(1+dydx^(-2))^0.5;
    dy=dy(:);
end
```

◆建模实验:导弹系统的改进

在命令行窗口执行下列脚本文件:

```
clear;close;
a=90;b=450;d=50;theta=pi/2;
[t,y]=ode45(@eg6_8fun,[0,0.1],[0 0],[],a,b,d,theta);
plot(y(:,1),y(:,2));
max(y(:,1)-d-a*t*cos(theta))
```

由于在T=0.1h内,$x(t)\geqslant d+atcos\theta$ 式均不成立,所以敌舰不在有效打击范围,应等近一些再发射.

◆ 建模实验:导弹系统的改进

(2) 离线算法 首先对于所有可能的d和θ,计算击中所需时间,从而对不同θ,得d的临界值. 具体应用时直接查表判断.

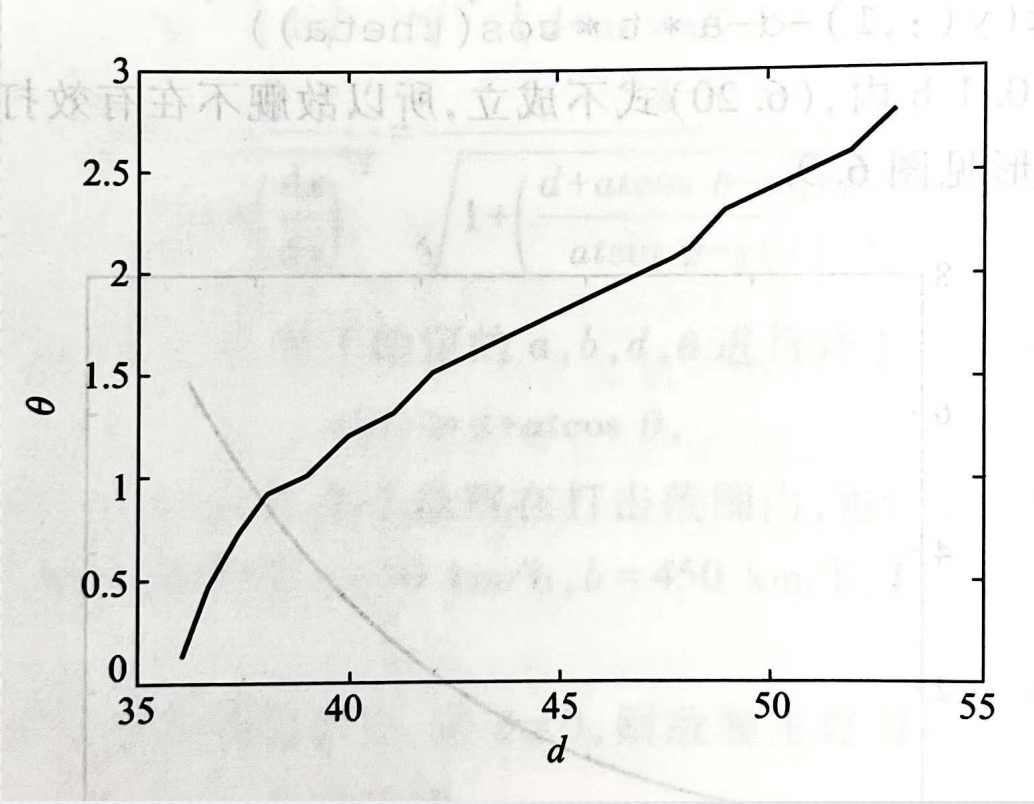
◆建模实验:导弹系统的改进

编写M脚本文件eg6_8b.m.

```
clear;close;
a=90;b=450;d=50;theta=pi/2;
i=1;
for d=54:-1:36
    for theta=0:0.1:pi,
        [t,y]=ode45(@eg6_8fun,[0,0.1],[0 0],[],a,b,d,theta);
        if max(y(:,1)-d-a*t*cos(theta))>0,
            range(i,:)=[d,theta];
            i=i+1;
            break;
        end
    end
end
plot(range(:,1),range(:,2));
xlabel('d');ylabel('theta');
```

◆ 建模实验:导弹系统的改进

运行得临界曲线，见图.



图中，曲线上方为打击范围. 由于0=1. 57，d=50在曲线下方，这样即可知不在打击范围内.

◆ 建模实验:导弹系统的改进

(3)计算机模拟 一个较基本但形象的方法.对于任意选定的参数a，b，d，θ，T，下面的M函数提供一个导弹追击敌舰演示工具.其中使用了MATLAB动画制作命令getframe和动画播放命令movie.

见Matlab程序文件 eg6_8c.m

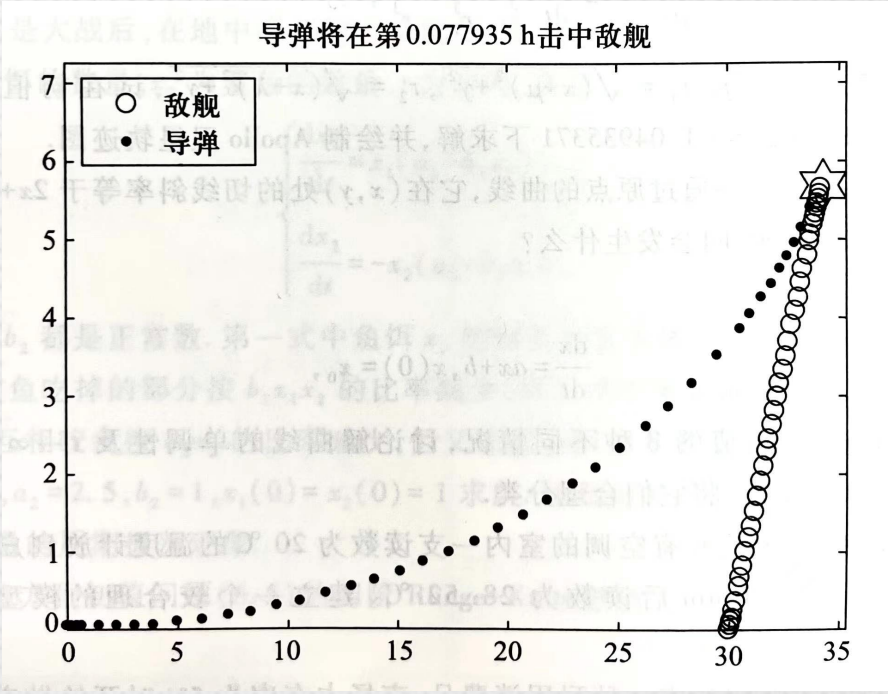
◆建模实验:导弹系统的改进

```
function m=eg6_8c(a,b,d,theta,T)
[t,y]=ode45(@eg6_8fun,[0,T],[0 0],[],a,b,d,theta);
x=[d+a*t*cos(theta),a*t* sin(theta)];
n=length(t);j=n;
for i=1:n
    plot(x(i,1),x(i,2),'o',y(i,1),y(i,2),'r.');
    axis([0 max(x(:,1)) 0 max(x(:,2))]);hold on;
    m(i)=getframe;
    if y(i,1)>=x(i,1)
        j=i;
        break;
    end
end
hold off; movie(m);
if j<n
    hold on;
    plot(y(j,1),y(j,2),'rh','markersize',18);
    hold off;
    title(['导弹将在第',num2str(t(j)),'h击中敌舰']);
else
    title(['导弹在',num2str(T),'h内不能击中敌舰']);
end
end
```

◆建模实验:导弹系统的改进

对于敌舰速度a=90km/h,导弹速度b=450km/h,距离d=30km,敌舰行驶角度$\theta = 0.3\pi$，反应时间T=0.1h，在命令行窗口执行>>eg6_8c(90,450,30,0.3*pi,0.1);

可见导弹约在t=0.08h时击中敌舰，位置约在(34,5.5).



◆ 建模实验:导弹系统的改进

总结:

应该说，三种算法各有千秋. 在线算法灵活，容易调整参数和模型，但速度慢；离线算法事先计算好，实时使用查询方式，不需计算，速度极快；模拟算法比较直观、生动.

实验4 应用性实验：地中海鲨鱼问题

应用性实验

(地中海鲨鱼问题)意大利生物学家D’Ancona曾致力于鱼类种群相互制约关系的研究。他从第一次世界大战期间地中海各港口捕获的几种鱼类捕获量百分比的资料中，发现鲨鱼等的比例有明显增加(下表)，而供其捕食的食用鱼的百分比却明显下降。显然战争使捕鱼量下降，食用鱼增加，鲨鱼等也随之增加，但为何鲨鱼的比例大幅增加呢?他无法解释这个现象，于是求助于著名的意大利数学家V.Volterra，希望能建立一个食饵—捕食系统的数学模型，定量回答这个问题。

年代	1914	1915	1916	1917	1918
百分比(%)	11.9	21.4	22.1	21.2	36.4
年代	1919	1920	1921	1922	1923
百分比(%)	27.3	16.0	15.9	14.8	19.7

◆应用性实验

下面介绍Volterra模型与实验求解。

1. 符号说明

$x_1(t)$:t时刻食饵的数量;

$x_2(t)$:t时刻捕食者的数量;

r_1:食饵独立生存时的增长率;

r_2:捕食者独立生存时的增长率;

λ_1:捕食者掠取食饵的能力;

λ_2:食饵对捕食者的供养能力;

e:人工捕获能力系数。

◆应用性实验

2. Volterra模型

模型1（不考虑人工捕获情形）：

$$\begin{cases} \dfrac{dx_1}{dt} = x_1(r_1 - \lambda_1 x_2) \\ \dfrac{dx_2}{dt} = x_2(-r_2 - \lambda_2 x_1) \end{cases}$$

对于数据$r_1 = 1, \lambda_1 = 0.1,\ r_2 = 0.5,\ \lambda_1 = 0.02,\ x_1(0) = 25,\ x_2(0) = 2$, t的终值取15（数值实验），上述模型为

$$\begin{cases} x'_1 = x_1(1 - 0.1x_2) \\ x'_2 = x_2(-0.5 + 0.02x_1) \\ x_1(0) = 25, x_2(0) = 2 \end{cases}$$

◆应用性实验

模型2（考虑人工捕获情形）：

e表示人工捕获能力系数，相当于食饵的自然增长率由r_1降为r_1-e，捕食者的死亡率由r_2增为r_2+e,从而模型为

$$\begin{cases}\dfrac{dx_1}{dt}=x_1[(r_1-e)-\lambda_1 x_2]\\ \dfrac{dx_2}{dt}=x_2[-(r_2+e)+\lambda_2 x_1]\end{cases}$$

◆应用性实验

仍取(模型1) $r_1 = 1, \lambda_1 = 0.1,\ r_2 = 0.5,\ \lambda_1 = 0.02,$ $x_1(0) = 25,\ x_2(0) = 2$, 设战前捕获能力系数e=0.3，战争中捕获能力系数e=0.1，则战前与战争中的模型分别为

$$\begin{cases} x'_1 = x_1(0.7 - 0.1x_2) \\ x'_2 = x_2(-0.8 + 0.02x_1) \\ x_1(0) = 25, x_2(0) = 2 \end{cases} \text{与} \begin{cases} x'_1 = x_1(0.9 - 0.1x_2) \\ x'_2 = x_2(-0.6 + 0.02x_1) \\ x_1(0) = 25, x_2(0) = 2 \end{cases}$$

◆应用性实验

3. 实验求解

1)模型1求解

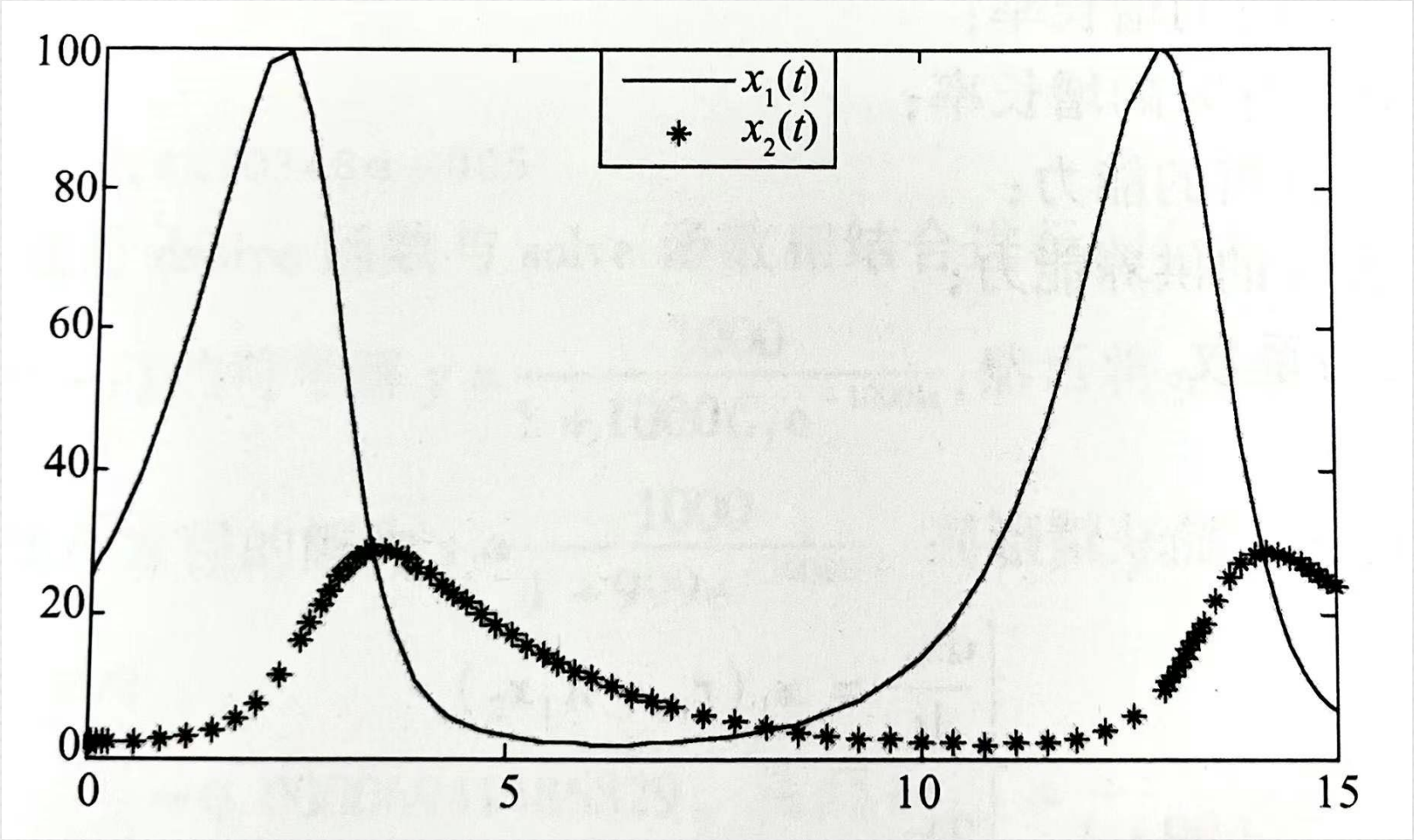
建立函数文件:

```
function xx=fun4_12(t,x)
xx=[x(1)*(1-0.1*x(2));x(2)*(-0.5+0.02*x(1))];
```

执行:

```
[t,x]=ode45(@fun4_12,[0,15],[25;2]);
plot(t,x(:,1),'-',t,x(:,2),'*');
legend('x1(t)','x2(t)');
```

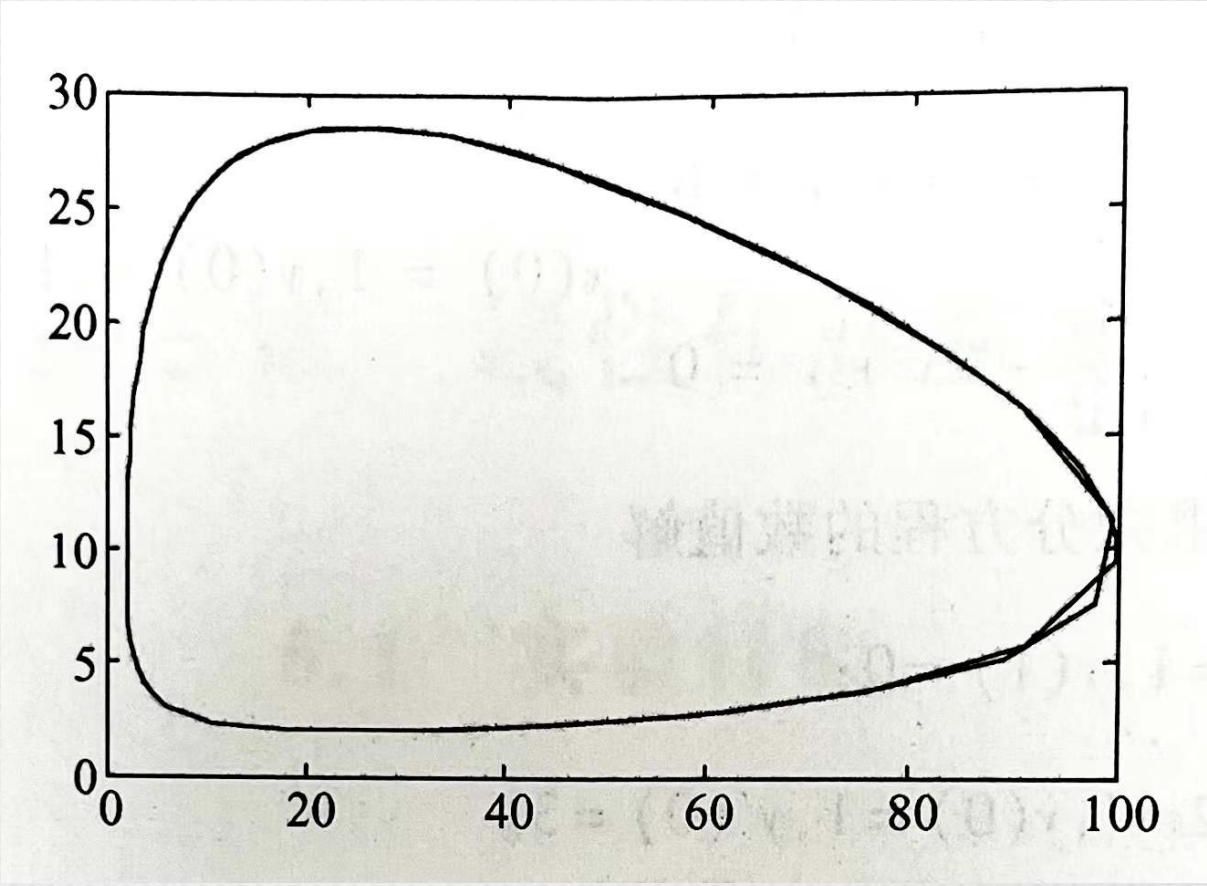
◆应用性实验



$x_1(t)$与$x_2(t)$关系图

◆应用性实验

```
>>figure
>>plot(x(:,1),x(:,2))
```



相图(x_1, x_2)

◆应用性实验

2)模型2求解

建立函数文件:

```
function xx=fun4_12e1(t,x)
xx=[x(1)*(0.7-0.1*x(2));x(2)*(-0.8+0.02*x(1))];
function xx=fun4_12e2(t,x)
xx=[x(1)*(0.9-0.1*x(2));x(2)*(-0.6+0.02*x(1))];
```

执行:

```
[t1,y]=ode45(@fun4_12e1,[0,15],[25;2]);
plot(t1,y(:,2)./(y(:,1)+y(:,2)),'-');
[t2,z]=ode45(@fun4_12e2,[0,15],[25;2]);
hold on
plot(t2,z(:,2)./(z(:,1)+z(:,2)),'*');
legend('战前','战争中');
hold off
```

◆应用性实验

程序执行结果如图所示，从图中可以看出：战争中的鲨鱼比例比战前高。