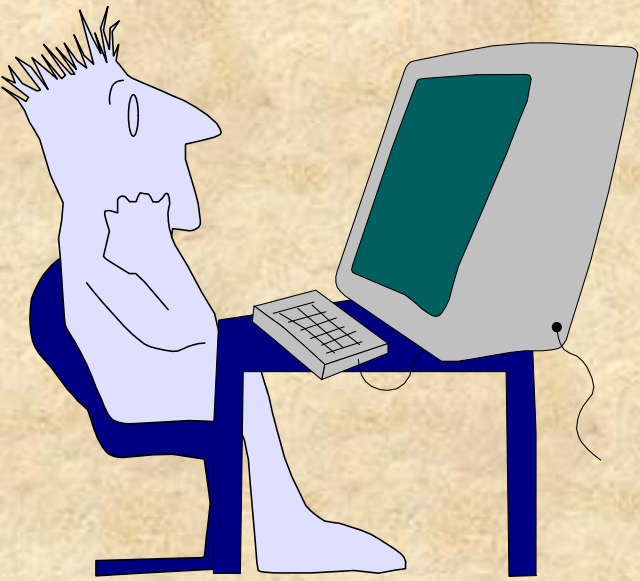


# 数学建模与数学实验

## MATLAB求解方程



1 线性方程组求解.

2 非线性方程数值求解.

3 函数极值.

4 求简单微分方程的解析解.

5 求微分方程的数值解.

6 数学建模实例.

7 实验作业.





# **MATLAB解方程与函数极值**

**(1) 线性方程组求解**

**(2) 非线性方程数值求解**

**(3) 函数极值**

# 求微分方程的数值解

(一) 常微分方程数值解的定义

(二) 建立数值解法的一些途径

(三) 用MATLAB软件求常微分方程的数值解

返回

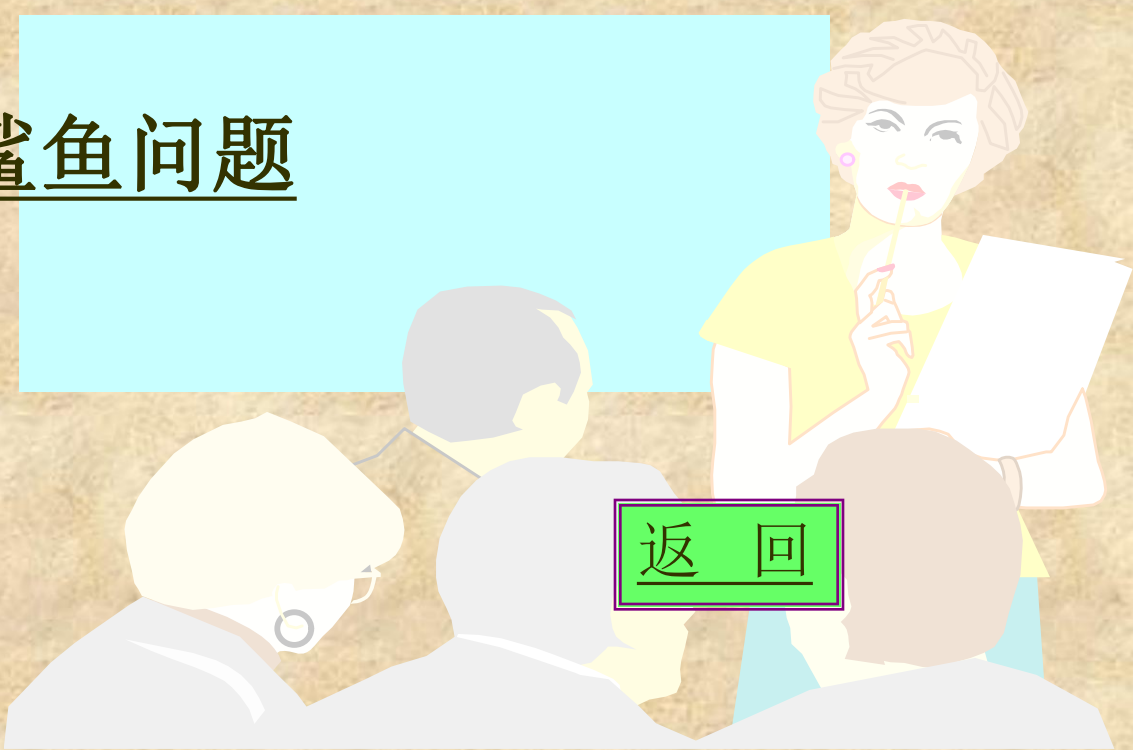


# 数学建模实例

1. 目标跟踪问题一：导弹追踪问题

2. 目标跟踪问题二：慢跑者与狗

3. 地中海鲨鱼问题



返回

## (1) 线性方程组求解

### 直接解法

利用左除运算符的直接解法

对于线性方程组  $Ax=b$ ，可以利用左除运算符 “\” 求解：

$$x=A\b$$



**eg1** 用直接解法求解下列线性方程组。

$$\begin{cases} 2x_1 + x_2 - 5x_3 + x_4 = 13, \\ x_1 - 5x_2 + 7x_4 = -9, \\ 2x_2 + x_3 - x_4 = 6, \\ x_1 + 6x_2 - x_3 - 4x_4 = 0. \end{cases}$$

**解：** 命令如下：

**A=[2,1,-5,1;1,-5,0,7;0,2,1,-1;1,6,-1,-4];**

**b=[13,-9,6,0]';**

**x=A\b**

## 利用矩阵的分解求解线性方程组

矩阵分解是指根据一定的原理用某种算法将一个矩阵分解成若干个矩阵的乘积。常见的矩阵分解有LU分解、QR分解、Cholesky分解，以及Schur分解、Hessenberg分解、奇异分解等。



## (1) LU分解

矩阵的LU分解就是将一个矩阵表示为一个交换下三角矩阵和一个上三角矩阵的乘积形式。线性代数中已经证明，只要方阵A是非奇异的，LU分解总是可以进行的。

MATLAB提供的lu函数用于对矩阵进行LU分解，其调用格式为：

$[L,U]=lu(X)$ ：产生一个上三角阵U和一个变换形式的下三角阵L(行交换)，使之满足 $X=LU$ 。注意，这里的矩阵X必须是方阵。

$[L,U,P]=lu(X)$ ：产生一个上三角阵U和一个下三角阵L以及一个置换矩阵P，使之满足 $PX=LU$ 。当然矩阵X同样必须是方阵。

实现LU分解后，线性方程组 $Ax=b$ 的解 $x=U\backslash(L\backslash b)$ 或 $x=U\backslash(L\backslash Pb)$ ，这样可以大大提高运算速度。



**eg2 用LU分解求解eg1中的线性方程组。**

$$\begin{cases} 2x_1 + x_2 - 5x_3 + x_4 = 13, \\ x_1 - 5x_2 + 7x_4 = -9, \\ 2x_2 + x_3 - x_4 = 6, \\ x_1 + 6x_2 - x_3 - 4x_4 = 0. \end{cases}$$

**命令如下：**

**A=[2,1,-5,1;1,-5,0,7;0,2,1,-1;1,6,-1,-4];**

**b=[13,-9,6,0]';**

**[L,U]=lu(A);**

**x=U\ (L\b)**

**或采用LU分解的第2种格式，命令如下：**

**[L,U,P]=lu(A);**

**x=U\ (L\ P\*b)**



## (2) QR分解

对矩阵 $X$ 进行QR分解，就是把 $X$ 分解为一个正交矩阵 $Q$ 和一个上三角矩阵 $R$ 的乘积形式。QR分解只能对方阵进行。MATLAB的函数`qr`可用于对矩阵进行QR分解，其调用格式为：

$[Q,R]=qr(X)$ ：产生一个正交矩阵 $Q$ 和一个上三角矩阵 $R$ ，使之满足 $X=QR$ 。

$[Q,R,E]=qr(X)$ ：产生一个正交矩阵 $Q$ 、一个上三角矩阵 $R$ 以及一个置换矩阵 $E$ ，使之满足 $XE=QR$ 。

实现QR分解后，线性方程组 $Ax=b$ 的解 $x=R\backslash(Q\backslash b)$ 或 $x=E(R\backslash(Q\backslash b))$ 。



**eg3 用QR分解求解例eg1中的线性方程组。**

$$\begin{cases} 2x_1 + x_2 - 5x_3 + x_4 = 13, \\ x_1 - 5x_2 + 7x_4 = -9, \\ 2x_2 + x_3 - x_4 = 6, \\ x_1 + 6x_2 - x_3 - 4x_4 = 0. \end{cases}$$

**命令如下：**

**A=[2,1,-5,1;1,-5,0,7;0,2,1,-1;1,6,-1,-4];**

**b=[13,-9,6,0]';**

**[Q,R]=qr(A);**

**x=R\((Q\b))**

**或采用QR分解的第2种格式，命令如下：**

**[Q,R,E]=qr(A);**

**x=E\*(R\((Q\b)))**



### (3) Cholesky分解

如果矩阵 $X$ 是对称正定的，则Cholesky分解将矩阵 $X$ 分解成一个下三角矩阵和上三角矩阵的乘积。设上三角矩阵为 $R$ ，则下三角矩阵为其转置，即 $X=R'R$ 。MATLAB函数chol(X)用于对矩阵 $X$ 进行Cholesky分解，其调用格式为：

$R=\text{chol}(X)$ ：产生一个上三角阵 $R$ ，使 $R'R=X$ 。若 $X$ 为非对称正定，则输出一个出错信息。

$[R,p]=\text{chol}(X)$ ：这个命令格式将不输出出错信息。当 $X$ 为对称正定的，则 $p=0$ ， $R$ 与上述格式得到的结果相同；否则 $p$ 为一个正整数。如果 $X$ 为满秩矩阵，则 $R$ 为一个阶数为 $q=p-1$ 的上三角阵，且满足 $R'R=X(1:q,1:q)$ 。

实现Cholesky分解后，线性方程组 $Ax=b$ 变成 $R'Rx=b$ ，所以 $x=R\backslash(R'\backslash b)$ 。



**eg4 用Cholesky分解求解eg1中的线性方程组。**

$$\begin{cases} 2x_1 + x_2 - 5x_3 + x_4 = 13, \\ x_1 - 5x_2 + 7x_4 = -9, \\ 2x_2 + x_3 - x_4 = 6, \\ x_1 + 6x_2 - x_3 - 4x_4 = 0. \end{cases}$$

**命令如下：**

**A=[2,1,-5,1;1,-5,0,7;0,2,1,-1;1,6,-1,-4];**

**b=[13,-9,6,0]';**

**R=chol(A)**

**??? Error using ==> chol**

**Matrix must be positive definite**

**命令执行时，出现错误信息，说明A为非正定矩阵。**



## 迭代解法

迭代解法非常适合求解大型系数矩阵的方程组。在数值分析中，迭代解法主要包括 Jacobi迭代法、Gauss-Seidel迭代法、超松弛迭代法和两步迭代法。

### 1. Jacobi迭代法

对于线性方程组  $Ax=b$ ，如果  $A$  为非奇异方阵，即  $a_{ii} \neq 0 (i=1,2,\dots,n)$ ，则可将  $A$  分解为  $A=D-L-U$ ，其中  $D$  为对角阵，其元素为  $A$  的对角元素， $L$  与  $U$  为  $A$  的下三角阵和上三角阵，于是  $Ax=b$  化为：

$$x=D^{-1}(L+U)x+D^{-1}b$$

与之对应的迭代公式为：

$$x(k+1)=D^{-1}(L+U)x(k)+D^{-1}b$$

这就是Jacobi迭代公式。如果序列  $\{x(k+1)\}$  收敛于  $x$ ，则  $x$  必是方程  $Ax=b$  的解。



**Jacobi迭代法的MATLAB函数文件Jacobi.m如下：**

```
function [y,n]=jacobi(A,b,x0,eps)  
if nargin==3  
    eps=1.0e-6;  
elseif nargin<3  
    error  
    return  
end  
D=diag(diag(A));    %求A的对角矩阵  
L=-tril(A,-1);    %求A的下三角阵  
U=-triu(A,1);    %求A的上三角阵  
B=D\(L+U);  
f=D\b;  
y=B*x0+f;  
n=1;    %迭代次数  
while norm(y-x0)>=eps  
    x0=y;  
    y=B*x0+f;  
    n=n+1;  
end
```



**eg5** 用Jacobi迭代法求解下列线性方程组。设迭代初值为0，迭代精度为 $10^{-6}$ 。

$$\begin{cases} 10x_1 - x_2 = 9, \\ -x_1 + 10x_2 - 2x_3 = 7, \\ -2x_2 + 10x_3 = 6. \end{cases}$$

在命令中调用函数文件Jacobi.m，命令如下：

```
A=[10,-1,0;-1,10,-2;0,-2,10];
```

```
b=[9,7,6]';
```

```
[x,n]=jacobi(A,b,[0,0,0]',1.0e-6)
```



## Gauss-Serdel迭代法

在Jacobi迭代过程中，计算时，已经得到，不必再用，即原来的迭代公式  $\mathbf{D} \mathbf{x}^{(k+1)} = (\mathbf{L} + \mathbf{U}) \mathbf{x}^{(k)} + \mathbf{b}$  可以改进为  $\mathbf{D} \mathbf{x}^{(k+1)} = \mathbf{L} \mathbf{x}^{(k+1)} + \mathbf{U} \mathbf{x}^{(k)} + \mathbf{b}$ ，于是得到：

$$\mathbf{x}^{(k+1)} = (\mathbf{D} - \mathbf{L})^{-1} \mathbf{U} \mathbf{x}^{(k)} + (\mathbf{D} - \mathbf{L})^{-1} \mathbf{b}$$

该式即为Gauss-Serdel迭代公式。和Jacobi迭代相比，Gauss-Serdel迭代用新分量代替旧分量，精度会高些。



**Gauss-Serdel迭代法的MATLAB函数文件gauseidel.m如下：**

```
function [y,n]=gauseidel(A,b,x0,eps)  
if nargin==3  
    eps=1.0e-6;  
elseif nargin<3  
    error  
    return  
end  
D=diag(diag(A));    %求A的对角矩阵  
L=-tril(A,-1);    %求A的下三角阵  
U=-triu(A,1);    %求A的上三角阵  
G=(D-L)\U;  
f=(D-L)\b;  
y=G*x0+f;  
n=1;    %迭代次数  
while norm(y-x0)>=eps  
    x0=y;  
    y=G*x0+f;  
    n=n+1;  
end
```



**eg6 用Gauss-Seidel迭代法求解eg5的线性方程组。设迭代初值为0，迭代精度为 $10^{-6}$ 。**

$$\begin{cases} 10x_1 - x_2 = 9, \\ -x_1 + 10x_2 - 2x_3 = 7, \\ -2x_2 + 10x_3 = 6. \end{cases}$$

**在命令中调用函数文件gauseidel.m，命令如下：**

**A=[10,-1,0;-1,10,-2;0,-2,10];**

**b=[9,7,6]';**

**[x,n]=gauseidel(A,b,[0,0,0]',1.0e-6)**



**eg7** 分别用Jacobi迭代和Gauss-Seidel迭代法求解下列线性方程组 $Ax=b$ ，看是否收敛。

$$\begin{cases} x_1 + 2x_2 - 2x_3 = 9, \\ x_1 + x_2 + x_3 = 7, \\ 2x_1 + 2x_2 + x_3 = 6. \end{cases}$$

命令如下：

```
A=[1,2,-2;1,1,1;2,2,1];
```

```
b=[9;7;6];
```

```
[x,n]=jacobi(A,b,[0;0;0])
```

```
[x,n]=gausidel(A,b,[0;0;0])
```



## (2) 非线性方程数值求解

### 单变量非线性方程求解

在MATLAB中提供了一个**fzero**函数，可以用来求单变量非线性方程的根。该函数的调用格式为：

**z=fzero('fname',x0,tol,trace)**

其中**fname**是待求根的函数文件名，**x0**为搜索的起点。一个函数可能有多个根，但**fzero**函数只给出离**x0**最近的那个根。**tol**控制结果的相对精度，缺省时取**tol=eps**，**trace**指定迭代信息是否在运算中显示，为1时显示，为0时不显示，缺省时取**trace=0**。



**eg8** 求  $f(x) = x - 10^x + 2 = 0$  在  $x_0 = 0.5$  附近的根。

步骤如下：

(1) 建立函数文件**funx.m**。

```
function fx=funx(x)
```

```
fx=x-10.^x+2;
```

(2) 调用**fzero**函数求根。

```
z=fzero('funx',0.5)
```

```
z =
```

```
0.3758
```



## 非线性方程组的求解

对于非线性方程组 $F(X)=0$ ，用fsolve函数求其数值解。  
fsolve函数的调用格式为：

**$X=fsolve('fun',X0,option)$**

其中X为返回的解，fun是用于定义需求解的非线性方程组的函数文件名，X0是求根过程的初值，option为最优化工具箱的选项设定。最优化工具箱提供了20多个选项，用户可以使用optimset命令将它们显示出来。如果想改变其中某个选项，则可以调用optimset()函数来完成。例如，Display选项决定函数调用时中间结果的显示方式，其中‘off’为不显示，‘iter’表示每步都显示，‘final’只显示最终结果。optimset(‘Display’,‘off’)将设定Display选项为‘off’。



eg9 求下列非线性方程组 
$$\begin{cases} x - 0.6 \sin x - 0.3 \cos y = 0 \\ y - 0.6 \cos x + 0.3 \sin y = 0 \end{cases}$$

在(0.5,0.5)附近的数值解。

(1) 建立函数文件myfun.m。

```
function q=myfun(p)
```

```
x=p(1);
```

```
y=p(2);
```

```
q(1)=x-0.6*sin(x)-0.3*cos(y);
```

```
q(2)=y-0.6*cos(x)+0.3*sin(y);
```

(2) 在给定的初值 $x_0=0.5, y_0=0.5$ 下，调用fsolve函数求方程的根。

```
x=fsolve('myfun',[0.5,0.5]',optimset('Display','off'))
```

```
x =
```

```
0.6354
```

```
0.3734
```



eg9 求下列非线性方程组 
$$\begin{cases} x - 0.6 \sin x - 0.3 \cos y = 0 \\ y - 0.6 \cos x + 0.3 \sin y = 0 \end{cases}$$

在(0.5,0.5)附近的数值解。

将求得的解代回原方程，可以检验结果是否正确，命令如下：

```
q=myfun(x)
```

```
q =
```

```
1.0e-009 *
```

```
0.2375  0.2957
```

可见得到了较高精度的结果。



### (3)函数极值

**MATLAB**提供了基于单纯形算法求解函数极值的函数 **fminbnd**，它们分别用于单变量函数的最小值，其调用格式为：

**x=fminbnd('fname',x1,x2)**

其中**fminbnd**函数用于求单变量函数的最小值点。**fname**是被最小化的目标函数名，**x1**和**x2**限定自变量的取值范围。

**MATLAB**没有专门提供求函数最大值的函数，但只要注意到 $-f(x)$ 在区间 $(a,b)$ 上的最小值就是 $f(x)$ 在 $(a,b)$ 的最大值，所以**fminbnd**( $-f,x1,x2$ )返回函数 $f(x)$ 在区间 $(x1,x2)$ 上的最大值。



eg10 求 $f(x)=x^3-2x-5$ 在 $[0,5]$ 内的最小值点。

(1) 建立函数文件mymin.m。

```
function fx=mymin(x)
```

```
fx=x.^3-2*x-5;
```

(2) 调用fmin函数求最小值点。

```
x=fminbnd('mymin',0,5)
```

```
x=
```

```
0.8165
```



# 微分方程的解析解

求微分方程（组）解析解的命令：

**dsolve**（'方程1'，'方程2'，...，'方程n'，'初始条件'，'自变量'）

记号：在表达微分方程时，用字母 D 表示求微分，D2、D3 等表示求高阶微分.任何 D 后所跟的字母为因变量，自变量可以指定或由系统规则选定为缺省.

例如，微分方程  $\frac{d^2 y}{dx^2} = 0$  应表达为：D2y=0.

例 1 求  $\frac{du}{dt} = 1 + u^2$  的通解.

解 输入命令：dsolve('Du=1+u^2','t')

结 果：u = tan(t+C<sub>2</sub>)

To MATLAB (ff1)



例 2 求微分方程的特解.

$$\begin{cases} \frac{d^2 y}{dx^2} + 4 \frac{dy}{dx} + 29y = 0 \\ y(0) = 0, y'(0) = 15 \end{cases}$$

解 输入命令:

```
y=dsolve('D2y+4*Dy+29*y=0','y(0)=0,Dy(0)=15','x')
```

To MATLAB (ff2)

结果为:  $y = 3e^{-2x}\sin(5x)$



例 3 求微分方程组的通解.

$$\begin{cases} \frac{dx}{dt} = 2x - 3y + 3z \\ \frac{dy}{dt} = 4x - 5y + 3z \\ \frac{dz}{dt} = 4x - 4y + 2z \end{cases}$$

解 输入命令：

```
[x,y,z]=dsolve('Dx=2*x-3*y+3*z',  
'Dy=4*x-5*y+3*z','Dz=4*x-4*y+2*z','t');  
x=simplify(x)           % 将x化简  
y=simplify(y)  
z=simplify(z)
```

结果为:  $x = c_1 e^{2t} + c_2 e^{-t}$

$$y = c_1 e^{2t} + c_2 e^{-t} + c_3 e^{-2t}$$

$$z = c_1 e^{2t} + c_3 e^{-2t}$$

To MATLAB (ff3)

返回



# 微分方程的数值解

## (一) 常微分方程数值解的定义

在生产和科研中所处理的微分方程往往很复杂,且大多得不出一般解. 而实际中的对初值问题,一般是要求得解在若干个点上满足规定精确度的近似值,或者得到一个满足精确度要求的便于计算的表达式.

因此, 研究常微分方程的数值解法是十分必要的.

对常微分方程 :  $\begin{cases} y' = f(x, y) \\ y(x_0) = y_0 \end{cases}$ , 其数值解是指由初始点 $x_0$  开始

的若干离散的 $x$ 处的值, 即对 $x_0 < x_1 < x_2 < \cdots < x_n$ , 求出准确值 $y(x_1)$ ,  $y(x_2), \cdots, y(x_n)$  的相应近似值 $y_1, y_2, \cdots, y_n$ .

[返回](#)



## (二) 建立数值解法的一些途径

设  $x_{i+1} - x_i = h$ ,  $i = 0, 1, 2, \dots, n-1$ , 则可用以下离散化方法求解

$$\text{微分方程} \quad \begin{cases} y' = f(x, y) \\ y(x_0) = y_0 \end{cases}$$

### 1. 用差商代替导数

若步长 $h$ 较小, 则有

$$y'(x) \approx \frac{y(x+h) - y(x)}{h}$$

故有公式:

$$\begin{cases} y_{i+1} = y_i + hf(x_i, y_i) \\ y_0 = y(x_0) \end{cases} \quad i = 0, 1, 2, \dots, n-1$$

此即欧拉法.



## 2. 使用数值积分

对方程 $y' = f(x, y)$ , 两边由 $x_i$ 到 $x_{i+1}$ 积分, 并利用梯形公式, 有:

$$\begin{aligned} y(x_{i+1}) - y(x_i) &= \int_{x_i}^{x_{i+1}} f(t, y(t)) dt \\ &\approx \frac{x_{i+1} - x_i}{2} [f(x_i, y(x_i)) + f(x_{i+1}, y(x_{i+1}))] \end{aligned}$$

故有公式:

$$\begin{cases} y_{i+1} = y_i + \frac{h}{2} [f(x_i, y_i) + f(x_{i+1}, y_{i+1})] \\ y_0 = y(x_0) \end{cases}$$

实际应用时, 与欧拉公式结合使用:

$$\begin{cases} y_{i+1}^{(0)} = y_i + hf(x_i, y_i) \\ y_{i+1}^{(k+1)} = y_i + \frac{h}{2} [f(x_i, y_i) + f(x_{i+1}, y_{i+1}^{(k)})] \quad k = 0, 1, 2, \dots \end{cases}$$

对于已给的精确度 $\varepsilon > 0$ , 当满足 $|y_{i+1}^{(k+1)} - y_{i+1}^{(k)}| < \varepsilon$  时, 取  $y_{i+1} = y_{i+1}^{(k+1)}$ ,

然后继续下一步  $y_{i+2}$  的计算.

此即改进的欧拉法.



### 3. 使用泰勒公式

以此方法为基础，有龙格-库塔法、线性多步法等方法.

### 4. 数值公式的精度

当一个数值公式的截断误差可表示为 $O(h^{k+1})$ （其中 $k$ 为正整数， $h$ 为步长）时，称它是一个 $k$ 阶公式.

$k$ 越大，则数值公式的精度越高.

- 欧拉法是一阶公式，改进的欧拉法是二阶公式.
- 龙格-库塔法有二阶公式和四阶公式.
- 线性多步法有四阶亚当斯外插公式和内插公式.

### (三) 用MATLAB软件求常微分方程的数值解

**[t, x]=solver('f',ts,x0,options)**

自变量值

函数值

ode45  
ode23  
ode113  
ode15s  
ode23s  
ode23t  
ode23tb

由待解  
方程写  
成的M  
文件名

ts=[t0,t  
f], t0、  
tf为自变  
量的初值  
和终值

函数的  
初  
值

**ode23:** 组合的2/3阶龙格—库塔—费尔贝格算法  
**ode45:** 运用组合的4/5阶龙格—库塔—费尔贝格算法

用于设定误差限(缺省时设定相对误差 $10^{-3}$ , 绝对误差 $10^{-6}$ ),  
命令为: `options=odeset('reltol',rt,'abstol',at)`,  
**rt, at:** 分别为设定的相对误差和绝对误差.



## MATLAB提供的常微分方程求解函数

函数	功能	函数	功能
ode45	中等阶次方法求解非病态微分方程	ode15i	可变阶次方法求解隐式微分方程
ode15s	可变阶次方法求解病态微分方程	decic	求解隐式微分方程初值问题
ode23	低阶次方法求解非病态微分方程	odextend	求常微分方程初值问题扩展解
ode113	可变阶次方法求解非病态微分方程	odeget	常微分方程可选参数
ode23t	梯形法求解中等病态微分方程	odeset	设置常微分方程可选参数
ode23tb	低阶次方法求解病态微分方程	deval	计算微分方程问题
ode23s	低阶次方法求解病态微分方程		



求解器	方法描述	使用场合
ode23	2到3阶Runge-Kutta算法，低精度	非刚性
ode45	4-5阶Runge-Kutta算法，中精度	非刚性
ode113	Adams算法，精度可到10的-3次到10^-4次	非刚性，计算时间比ode45快
ode23t	梯形算法	适度刚性
ode15s	反向数值微分算法，中精度	刚性
ode23s	2阶Rosebrock算法，低精度	刚性，当精度较低时，计算时间比ode15s短
ode23tb	梯形算法，低精度	刚性，当精度较低时，计算时间比ode15s短
ode15i	可变秩求法	完全隐形微分方程



注意:

1. 在解含 $n$ 个未知数的方程组时,  $x_0$ 和 $x$ 均为 $n$ 维向量, M文件中的待解方程组应以 $x$ 的分量形式写出.
2. 使用MATLAB软件求数值解时, 高阶微分方程必须等价地变换成一阶微分方程组.

例 4 
$$\begin{cases} \frac{d^2x}{dt^2} - 1000(1-x^2)\frac{dx}{dt} + x = 0 \\ x(0) = 2; x'(0) = 0 \end{cases}$$

解：令  $y_1 = x, y_2 = y_1'$

则微分方程变为一阶微分方程组：

$$\begin{cases} y_1' = y_2 \\ y_2' = 1000(1-y_1^2)y_2 - y_1 \\ y_1(0) = 2, y_2(0) = 0 \end{cases}$$

1. 建立M文件vdp1000.m如下：

```
function dy=vdp1000(t,y)
```

```
dy=zeros(2,1);
```

```
dy(1)=y(2);
```

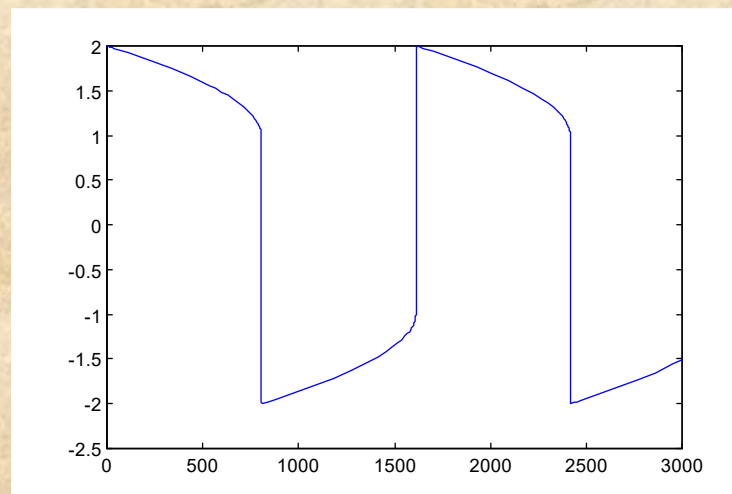
```
dy(2)=1000*(1-y(1)^2)*y(2)-y(1);
```

2. 取 $t_0=0$ ,  $t_f=3000$ , 输入命令：

```
[T,Y]=ode15s('vdp1000',[0 3000],[2 0]);
```

```
plot(T,Y(:,1),'-')
```

3. 结果如图



**To MATLAB (ff4)**



## 例 5 解微分方程组.

$$\begin{cases} y_1' = y_2 y_3 \\ y_2' = -y_1 y_3 \\ y_3' = -0.51 y_1 y_2 \\ y_1(0) = 0, y_2(0) = 1, y_3(0) = 1 \end{cases}$$

解

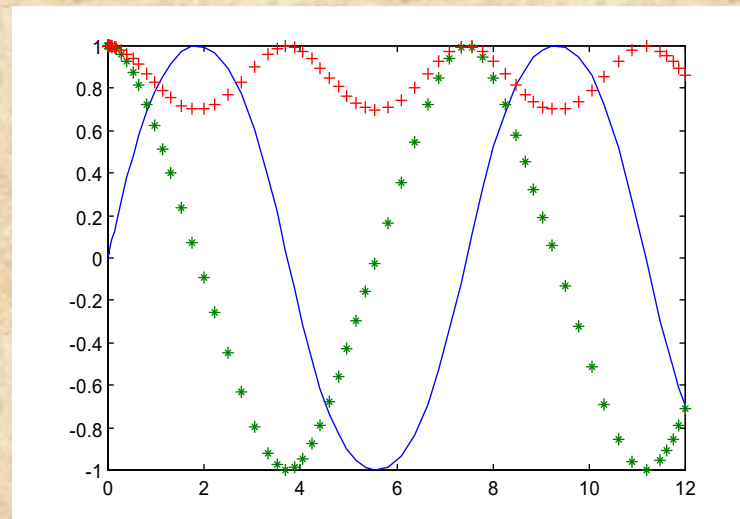
1. 建立M文件rigid. m如下:

```
function dy=rigid(t,y)
dy=zeros(3,1);
dy(1)=y(2)*y(3);
dy(2)=-y(1)*y(3);
dy(3)=-0.51*y(1)*y(2);
```

2. 取 $t_0=0$ ,  $t_f=12$ , 输入命令:

```
[T,Y]=ode45('rigid',[0 12],[0 1 1]);
plot(T,Y(:,1),'-',T,Y(:,2),'*',T,Y(:,3),'+')
```

3. 结果如图



To MATLAB (ff5)

返 回

图中,  $y_1$ 的图形为实线,  $y_2$ 的图形为“\*”线,  $y_3$ 的图形为“+”线.

## 例 6 解微分方程组.

$$\begin{cases} y_1' = y_2 y_3 \\ y_2' = -y_1 y_3 \\ y_3' = -0.51 y_1 y_2 \\ y_1(0) = 0, y_2(0) = 1, y_3(0) = 1 \end{cases}$$

解

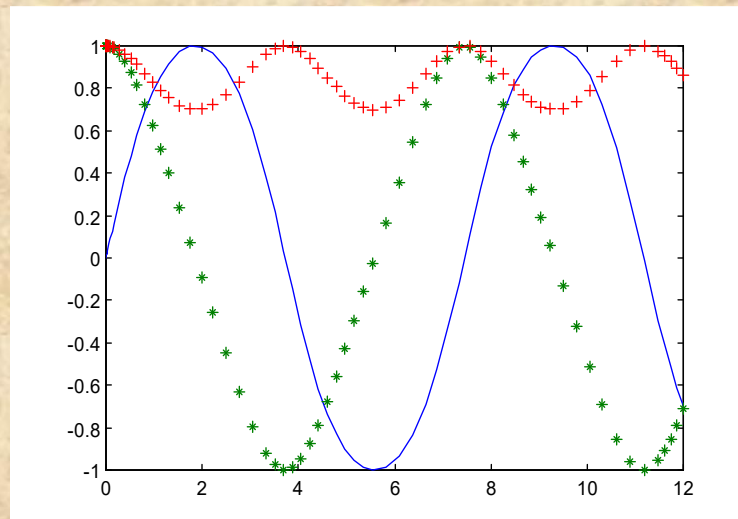
1. 建立M文件rigid. m如下:

```
function dy=rigid(t,y)
dy=zeros(3,1);
dy(1)=y(2)*y(3);
dy(2)=-y(1)*y(3);
dy(3)=-0.51*y(1)*y(2);
```

2. 取 $t_0=0$ ,  $t_f=12$ , 输入命令:

```
[T,Y]=ode45('rigid',[0 12],[0 1 1]);
plot(T,Y(:,1),'-',T,Y(:,2),'*',T,Y(:,3),'+')
```

3. 结果如图



[To MATLAB \(ff5\)](#)

[返回](#)

图中,  $y_1$ 的图形为实线,  $y_2$ 的图形为“\*”线,  $y_3$ 的图形为“+”线.





## 导弹追踪问题

设位于坐标原点的甲舰向位于 $x$ 轴上点 $A(1, 0)$ 处的乙舰发射导弹，导弹头始终对准乙舰. 如果乙舰以最大的速度 $v_0$ (常数)沿平行于 $y$ 轴的直线行驶，导弹的速度是 $5v_0$ ，求导弹运行的曲线方程. 乙舰行驶多远时，导弹将它击中？

### 解法一（解析法）

假设 $t$ 时刻导弹的位置为 $P(x(t), y(t))$ ，乙舰位于 $Q(1, v_0 t)$ .

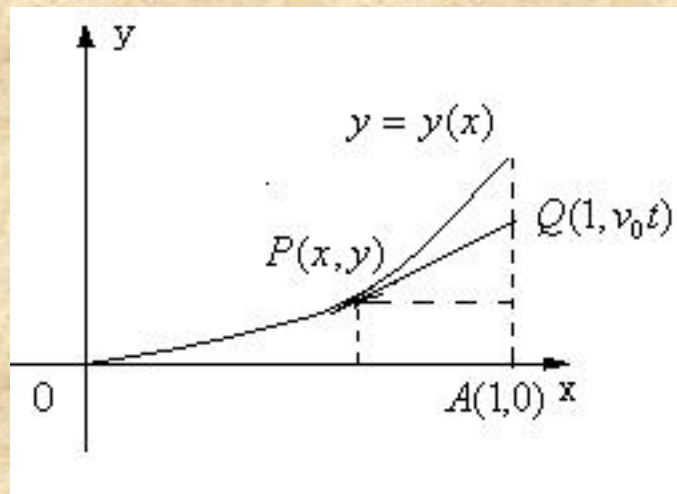
由于导弹头始终对准乙舰，故此时直线 $PQ$ 就是导弹的轨迹曲线弧 $OP$ 在点 $P$ 处的切线，

即有 
$$y' = \frac{v_0 t - y}{1 - x}$$

即 
$$v_0 t = (1 - x)y' + y \quad (1)$$

又根据题意，弧 $OP$ 的长度为 $|AQ|$ 的5倍，

即 
$$\int_0^x \sqrt{1 + y'^2} dx = 5v_0 t \quad (2)$$





由(1),(2)消去 $t$ , 整理得模型:

$$(1-x)y'' = \frac{1}{5}\sqrt{1+y'^2} \quad (3)$$

初值条件为:  $y(0) = 0$      $y'(0) = 0$

其解即为导弹的运行轨迹:

$$y = -\frac{5}{8}(1-x)^{\frac{4}{5}} + \frac{5}{12}(1-x)^{\frac{6}{5}} + \frac{5}{24}$$

当  $x = 1$  时  $y = \frac{5}{24}$ , 即当乙舰航行到点  $(1, \frac{5}{24})$  处时被导弹击中.

被击中时间为:  $t = \frac{y}{v_0} = \frac{5}{24v_0}$ . 若  $v_0=1$ , 则在  $t=0.21$  处被击中.

轨迹图见程序chase1

To  
MATLAB(chase1)



## 解法二(数值解法)

令 $y_1=y, y_2=y_1'$ ，将方程（3）化为一阶微分方程组.

$$(1-x)y'' = \frac{1}{5}\sqrt{1+y'^2} \quad \Rightarrow \quad \begin{cases} y_1' = y_2 \\ y_2' = \frac{1}{5}\sqrt{1+y_1^2} / (1-x) \end{cases}$$

1. 建立M文件eq1. m

```
function dy=eq1(x,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=1/5*sqrt(1+y(1)^2)/(1-x);
```

2. 取 $x_0=0, x_f=0.9999$ ，建立主程序ff6. m如下：

```
x0=0, xf=0.9999
[x,y]=ode15s('eq1',[x0 xf],[0 0]);
plot(x,y(:,1),'g. ')
hold on
y=0:0.01:2;
plot(1,y,'b*')
```

To MATLAB(ff6)

结论：导弹大致在（1，0.2）处击中乙舰.



### 解法三(建立参数方程求数值解)

设时刻 $t$ 乙舰的坐标为 $(X(t), Y(t))$ , 导弹的坐标为 $(x(t), y(t))$ .

1. 设导弹速度恒为  $w$ , 则  $\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 = w^2$  (1)

2. 由于弹头始终对准乙舰, 故导弹的速度平行于乙舰与导弹头位置的差向量,

即: 
$$\begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{pmatrix} = \lambda \begin{pmatrix} X - x \\ Y - y \end{pmatrix}, \quad \lambda > 0$$
 (2)

消去  $\lambda$  得: 
$$\begin{cases} \frac{dx}{dt} = \frac{w}{\sqrt{(X-x)^2 + (Y-y)^2}} (X-x) \\ \frac{dy}{dt} = \frac{w}{\sqrt{(X-x)^2 + (Y-y)^2}} (Y-y) \end{cases}$$
 (3)

3. 因乙舰以速度 $v_0$ 沿直线 $x=1$ 运动, 设 $v_0=1$ , 则 $w=5$ ,  $X=1$ ,  $Y=t$ .



因此导弹运动轨迹的参数方程为：

$$\begin{cases} \frac{dx}{dt} = \frac{5}{\sqrt{(1-x)^2 + (t-y)^2}} (1-x) \\ \frac{dy}{dt} = \frac{5}{\sqrt{(1-x)^2 + (t-y)^2}} (t-y) \\ x(0) = 0, y(0) = 0 \end{cases}$$

#### 4. 解导弹运动轨迹的参数方程

建立M文件eq2. m如下：

```
function dy=eq2(t,y)
dy=zeros(2,1);
dy(1)=5*(1-y(1))/sqrt((1-y(1))^2+(t-y(2))^2);
dy(2)=5*(t-y(2))/sqrt((1-y(1))^2+(t-y(2))^2);
```

取t0=0，tf=2，建立主程序chase2. m如下：

```
[t,y]=ode45('eq2',[0 2],[0 0]);
Y=0:0.01:2;
plot(1,Y,'-'), hold on
plot(y(:,1),y(:,2),'*')
```

To MATLAB(chase2)

## 5. 结果见图1

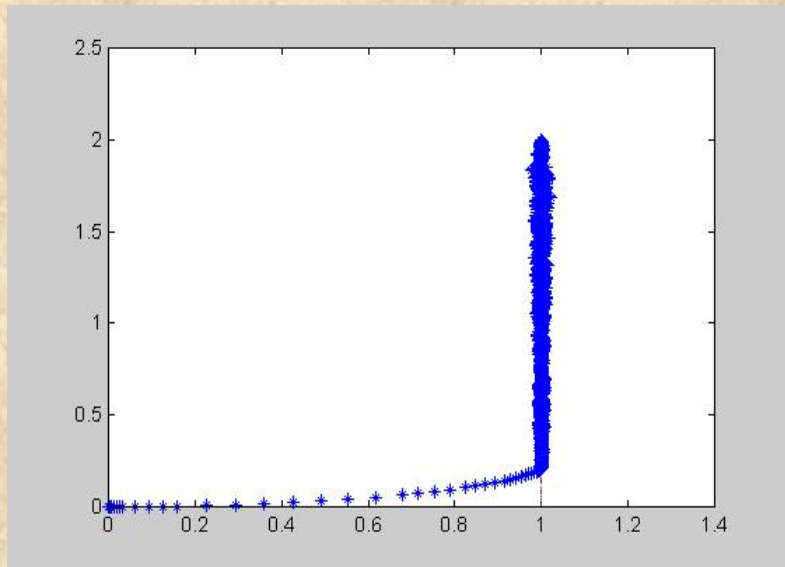


图1

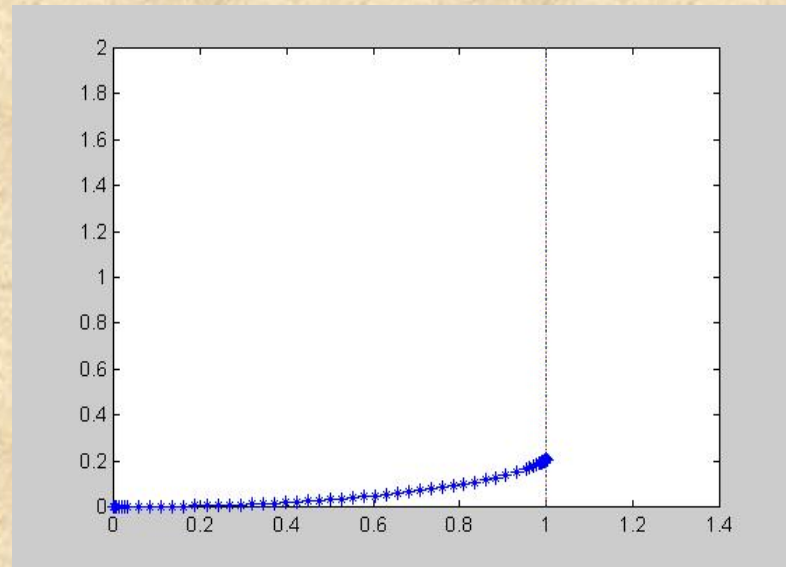


图2

导弹大致在  $(1, 0.2)$  处击中乙舰，与前面的结论一致. 在 `chase2.m` 中，按二分法逐步修改  $t_f$ ，即分别取  $t_f=1, 0.5, 0.25, \dots$ ，直到  $t_f=0.21$  时，得图2.

[To MATLAB\(chase2\)](#)

结论：时刻  $t=0.21$  时，导弹在  $(1, 0.21)$  处击中乙舰. [返回](#)





## 慢跑者与狗

一个慢跑者在平面上沿椭圆以恒定的速率 $v=1$ 跑步,设椭圆方程为:  $x=10+20\cos t$ ,  $y=20+15\sin t$ . 突然有一只狗攻击他. 这只狗从原点出发,以恒定速率 $w$ 跑向慢跑者,狗的运动方向始终指向慢跑者. 分别求出 $w=20, w=5$ 时狗的运动轨迹.



### 1. 模型建立

设 $t$ 时刻慢跑者的坐标为 $(X(t), Y(t))$ , 狗的坐标为 $(x(t), y(t))$ 则  $X=10+20\cos t$ ,  $Y=20+15\sin t$ . 狗从 $(0,0)$ 出发, 与导弹追踪问题类似, 狗的运动轨迹的参数方程为:

$$\begin{cases} \frac{dx}{dt} = \frac{w}{\sqrt{(10+20\cos t-x)^2 + (20+15\sin t-y)^2}} (10+20\cos t-x) \\ \frac{dy}{dt} = \frac{w}{\sqrt{(10+20\cos t-x)^2 + (20+15\sin t-y)^2}} (20+15\sin t-y) \\ x(0) = 0, \quad y(0) = 0 \end{cases}$$



## 2. 模型求解

(1)  $w=20$ 时,建立M文件eq3. m如下:

```
function dy=eq3(t,y)
dy=zeros(2,1);
dy(1)=20*(10+20*cos(t)-y(1))/sqrt
    ((10+20*cos(t)-y(1))^2+(20+15*sin(t)-y(2))^2);
dy(2)=20*(20+15*sin(t)-y(2))/sqrt
    ((10+20*cos(t)-y(1))^2+(20+15*sin(t)-y(2))^2);
```

取 $t_0=0$ ,  $t_f=10$ , 建立主程序chase3. m如下:

```
t0=0;tf=10;
[t,y]=ode45('eq3',[t0 tf],[0 0]);
T=0:0.1:2*pi;
X=10+20*cos(T);
Y=20+15*sin(T);
plot(X,Y,'-')
hold on
plot(y(:,1),y(:,2),'*')
```

To MATLAB(chase3)

在chase3. m中, 不断修改 $t_f$ 的值, 分别取 $t_f=5, 2.5, 3.5, \dots$ , 至3.15时, 狗刚好追上慢跑者.



## (2) $w=5$ 时

建立M文件eq4. m如下:

```
function dy=eq4(t,y)
dy=zeros(2,1);
dy(1)=5*(10+20*cos(t)-y(1))/sqrt
    ((10+20*cos(t)-y(1))^2+(20+15*sin(t)-y(2))^2);
dy(2)=5*(20+15*sin(t)-y(2))/sqrt
    ((10+20*cos(t)-y(1))^2+(20+15*sin(t)-y(2))^2);
```

取 $t_0=0$ ,  $t_f=10$ , 建立主程序chase4. m如下:

```
t0=0;tf=10;
[t,y]=ode45('eq4',[t0 tf],[0 0]);
T=0:0.1:2*pi;
X=10+20*cos(T);
Y=20+15*sin(T);
plot(X,Y,'-')
hold on
plot(y(:,1),y(:,2),'*')
```

To MATLAB(chase4)

在chase3. m中, 不断修改 $t_f$ 的值, 分别取 $t_f=20, 40, 80, \dots$ , 可以看出, 狗永远追不上慢跑者.

[返回](#)



## 地中海鲨鱼问题

意大利生物学家Ancona曾致力于鱼类种群相互制约关系的研究，从第一次世界大战期间，地中海各港口几种鱼类捕获量百分比的资料中，他发现鲨鱼等的比例有明显增加（见下表），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，从而食用鱼增加，鲨鱼等也随之增加，但为何鲨鱼的比例大幅增加呢？

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

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，希望建立一个食饵—捕食系统的数学模型，定量地回答这个问题。



## 1. 符号说明:

$x_1(t)$ ——食饵在  $t$  时刻的数量;  $x_2(t)$ ——捕食者在  $t$  时刻的数量;

$r_1$ ——食饵独立生存时的增长率;  $r_2$ ——捕食者独自存在时的死亡率;

$\lambda_1$ ——捕食者掠取食饵的能力;  $\lambda_2$ ——食饵对捕食者的供养能力.

$e$ ——捕获能力系数

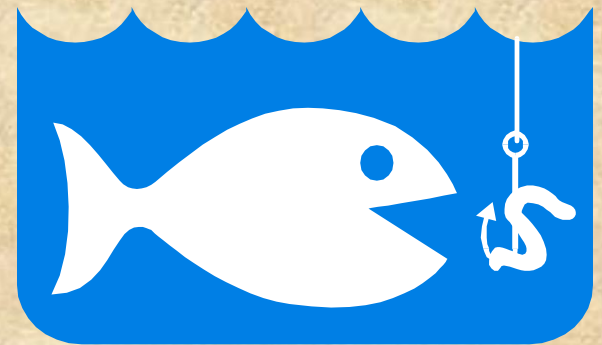
## 2. 基本假设:

- (1) 由于捕食者的存在使食饵增长率降低, 假设降低的程度与捕食者数量成正比;
- (2) 捕食者由于食饵为它提供食物, 其死亡率降低或增长, 假定增长的程度与食饵数量成正比.

## 3. 模型建立与求解

模型 (一) 不考虑人工捕获

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



该模型反映了在没有人工捕获的自然环境中食饵与捕食者之间的制约关系, 没有考虑食饵和捕食者自身的阻滞作用, 是Volterra提出的最简单的模型.



针对一组具体的数据用 MATLAB 软件进行计算.

设食饵和捕食者的初始数量分别为  $x_1(0) = x_{10}$ ,  $x_2(0) = x_{20}$

对于数据  $r_1 = 1, \lambda_1 = 0.1, r_2 = 0.5, \lambda_2 = 0.02, x_{10} = 25, x_{20} = 2$ ,

$t$  的终值经试验后确定为 15, 即模型为:

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

首先, 建立M文件shier. m如下:

```
function dx=shier(t,x)
dx=zeros(2,1);
dx(1)=x(1)*(1-0.1*x(2));
dx(2)=x(2)*(-0.5+0.02*x(1));
```

其次, 建立主程序shark. m如下:

```
[t,x]=ode45('shier',[0 15],[25 2]);
plot(t,x(:,1),'-',t,x(:,2),'*')
plot(x(:,1),x(:,2))
```

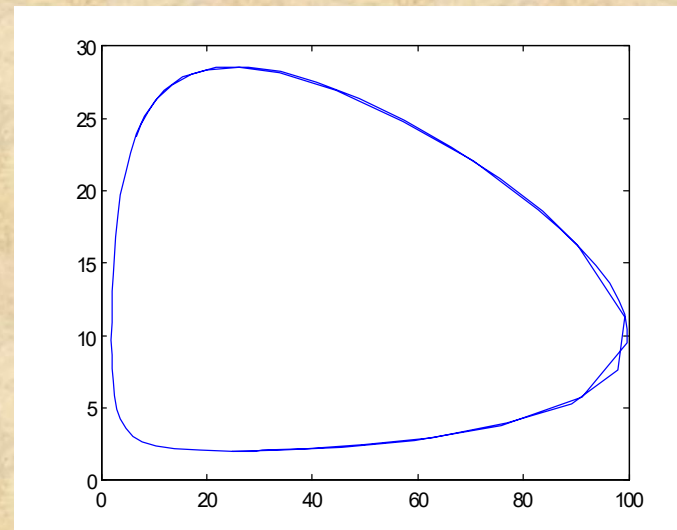
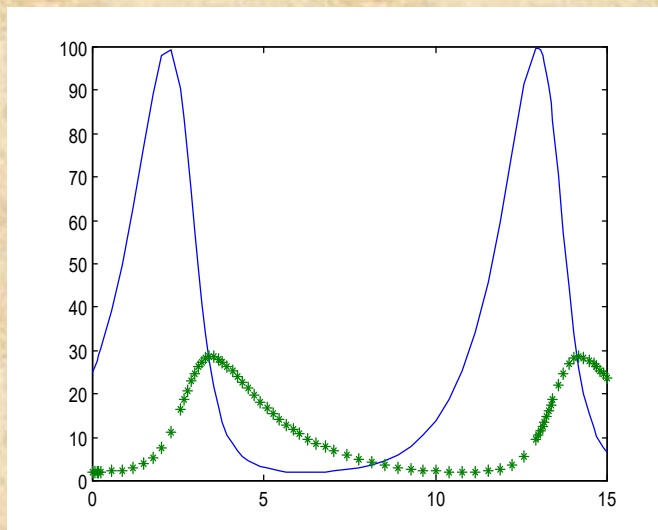


To MATLAB(shark)



求解结果：

数值解如下图： $x_1(t)$ 为实线， $x_2(t)$ 为“\*”线. 相图 $(x_1, x_2)$ 为：



左图反映了 $x_1(t)$ 与 $x_2(t)$ 的关系.

可以猜测： $x_1(t)$ 与 $x_2(t)$ 都是周期函数.

## 模型（二） 考虑人工捕获

设表示捕获能力的系数为 $e$ ，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 $r_2+e$

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

仍取 $r_1 = 1, \lambda_1 = 0.1, r_2 = 0.5, \lambda_2 = 0.02, x_1(0) = 25, x_2(0) = 2$

设战前捕获能力系数 $e=0.3$ ，战争中降为 $e=0.1$ ，则战前与战争中的模型分别为：

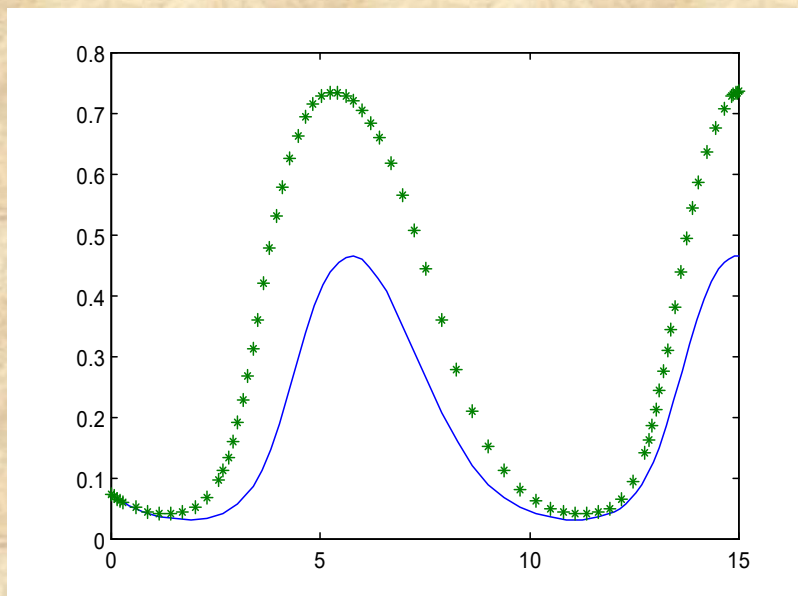
$$\begin{cases} \frac{dx_1}{dt} = x_1(0.7 - 0.1x_2) \\ \frac{dx_2}{dt} = x_2(-0.8 + 0.02x_1) \\ x_1(0) = 25, x_2(0) = 2 \end{cases}$$

$$\begin{cases} \frac{dx_1}{dt} = x_1(0.9 - 0.1x_2) \\ \frac{dx_2}{dt} = x_2(-0.6 + 0.02x_1) \\ x_1(0) = 25, x_2(0) = 2 \end{cases}$$



模型求解:

1. 分别用M文件shier1. m和shier2. m定义上述两个方程.
2. 建立主程序shark1. m求解两方程, 并画出两种情况下鲨鱼数在鱼类总数中所占比例  $x_2(t)/[x_1(t)+x_2(t)]$  的图形.



To MATLAB(shark1)

实线为战前的鲨鱼比例, “\*” 线为战争中的鲨鱼比例

结论: 战争中鲨鱼的比例比战前高!

[返回](#)

## 课后问题

1. 求微分方程的解析解，并画出图形，

$$y'=y+2x, y(0)=1, 0<x<1.$$

2. 求微分方程的数值解并画出图形，

$$y''+y\cos x=0, y(0)=1, y'(0)=0$$

3. 两种相似的群体之间为了争夺有限的同一种食物来源和生活空间而进行生存竞争时，往往是竞争力较弱的种群灭亡，而竞争力较强的种群达到环境容许的最大数量。

假设有甲、乙两个生物种群，当它们各自生存于一个自然环境中，均服从Logistic规律。



- (1)  $x_1(t), x_2(t)$  是两个种群的数量;
- (2)  $r_1, r_2$  是它们的固有增长率;
- (3)  $n_1, n_2$  是它们的最大容量;
- (4)  $m_2(m_1)$  为种群乙 (甲) 占据甲 (乙) 的位置的数量, 并且  $m_1 = \alpha x_2, m_2 = \beta x_1$ 。

$$\begin{cases} \frac{dx_1}{dt} = r_1 x_1 \left(1 - \frac{x_1 + m_2}{n_1}\right) \\ \frac{dx_2}{dt} = r_2 x_2 \left(1 - \frac{x_2 + m_1}{n_2}\right) \end{cases}$$

(1) 设  $r_1 = r_2 = 1, n_1 = n_2 = 100, m_1 = 0.5, m_2 = 2, x_{10} = x_{20} = 10$ 。计算  $x_1(t), x_2(t)$ , 画出图形及相轨迹图。解释其解变化过程。

(2) 设  $r_1 = r_2 = 1, n_1 = n_2 = 100, x_{10} = x_{20} = 10, \alpha = 1.5, \beta = 0.7$ , 计算  $x_1(t), x_2(t)$ , 画出图形及相轨迹图。解释其解变化过程。