



数学软件 Matlab

—— 多项式运算

—— 代数方程求解

内容提要

■ 多项式运算

- 多项式转化为符号表达式: `poly2sym`
- 四则运算: `conv`、`deconv`
- 导数与积分: `polyder`、`polyint`
- 求值与零点: `polyval`、`polyvalm`、`roots`、`poly`

■ 代数方程求解

- 线性方程组数值求解: `linsolve`
- 非线性方程数值求解: `fzero`
- 非线性方程符号求解: `solve`

多项式表示方法

- Matlab 中多项式的表示方法

- 在 Matlab 中，n 次多项式用一个长度为 n+1 的向量来表示

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

在 Matlab 中表示为向量： $[a_n, a_{n-1}, \dots, a_1, a_0]$

例： $2x^3 - x^2 + 3 \longleftrightarrow [2, -1, 0, 3]$

注：系数中的零不能省！

- 多项式与符号表达式的互化：`poly2sym`，`sym2poly`

例：`poly2sym([2, -1, 0, 3])`

多项式加减

● 多项式加减运算

- Matlab 没有提供专门进行多项式加减运算的函数
- 多项式的加减就是其所对应的**系数向量的加减运算**

- 次数相同的多项式，可直接对其系数向量进行加减运算
- 如果两个多项式次数不同，则应该把低次多项式中系数不足的高次项用 0 补足，然后再进行加减运算

例：

$$p_1 = 2x^3 - x^2 + 3 \quad \longrightarrow \quad [2, -1, 0, 3]$$
$$p_2 = \quad \quad \quad 2x + 1 \quad \longrightarrow \quad [0, 0, 2, 1]$$
$$p_1 + p_2 = 2x^3 - x^2 + 2x + 4 \quad \longleftarrow \quad [2, -1, 2, 4]$$

多项式乘除

- 多项式乘法运算: $k = \text{conv}(p, q)$

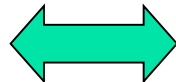
例: 计算多项式 $2x^3 - x^2 + 3$ 和 $2x + 1$ 的乘积

```
p=[2, -1, 0, 3];  
q=[2, 1];  
k=conv(p, q)
```

- 多项式除法运算: $[k, r] = \text{deconv}(p, q)$

- 其中 k 返回的是多项式 p 除以 q 的商, r 是余式

$$[k, r] = \text{deconv}(p, q)$$



$$p = \text{conv}(q, k) + r$$

多项式求导

- 多项式求导: `polyder`

<code>k=polyder(p)</code>	多项式 <code>p</code> 的导数
<code>k=polyder(p,q)</code>	<code>p*q</code> 的导数
<code>[k,d]=polyder(p,q)</code>	<code>p/q</code> 的导数, <code>k</code> 是分子, <code>d</code> 是分母

例: 已知 $p_1(x) = 2x^3 - x^2 + 3$, $p_2(x) = 2x + 1$

求: p_1' , $(p_1 p_2)'$, $(p_1 / p_2)'$

```
k1=polyder([2,-1,0,3]);  
k2=polyder([2,-1,0,3],[2,1]);  
[k3,d]=polyder([2,-1,0,3],[2,1]);
```

多项式积分

- 多项式积分: `polyint`

<code>I=polyint(p,c)</code>	不定积分, 常数项取 <code>c</code>
<code>I=polyint(p)</code>	不定积分, 常数项取 <code>0</code>

例: 已知 $p(x) = 2x^3 - x^2 + 3$

求 $\int p(x) dx$, 常数项取 5

```
I=polyint([2,-1,0,3],5);
```

多项式求值

- 多项式求值: `polyval`

<code>y=polyval(p,x)</code>	计算多项式 <code>p</code> 在 <code>x</code> 点的值
-----------------------------	---

- 这里的 `x` 可以是向量或矩阵, 此时采用的是数组运算!

例: 已知 $p(x) = 2x^3 - x^2 + 3$, 计算 p 在 x 和 y 的每个分量上的值, 其中 $x=2$, $y=[-1,2; -2,1]$

```
p=[2, -1, 0, 3];  
x=2;  
y=[ -1, 2; -2, 1];  
z1=polyval(p,x)  
z2=polyval(p,y)
```


矩阵多项式求值

- 矩阵多项式求值：`polyvalm`

`Y=polyvalm(p,A)`

计算多项式 `p` 作用在矩阵 `A` 上的值

- 这里的 `A` 必须是方阵，采用的是普通矩阵运算！

例：已知 $p(x) = 2x^3 - x^2 + 3$ ，则

```
polyvalm(p,A)=2*A*A*A - A*A + 3*eye(size(A))
```

```
polyval(p,A) = 2*A.*A.*A - A.*A + 3*ones(size(A))
```

```
p=[1,0,0]; % p(x)=x^2
```

```
x=[1, 2; 3, 4];
```

```
y1=polyval(p,x)
```

```
y2=polyvalm(p,x)
```

多项式的零点

- 多项式的零点：`roots`

`x=roots(p)`

计算多项式 `p` 的所有零点

- 这里的 `x` 是由 `p` 的所有零点组成的向量

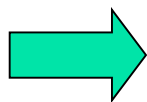
例：已知 $p(x) = 2x^3 - x^2 + 3$ ，求 $p(x)$ 的零点

`p=[2, -1, 0, 3];`

`x=roots(p)`

- 若已知多项式的全部零点，则可用 `poly` 函数给出该多项式

`p=poly(x)`



$p(x) = (x - x_1)(x - x_2) \cdots (x - x_n)$

多项式运算小结

```
k=conv(p,q)  
[k,r]=deconv(p,q)
```

```
k=polyder(p)  
k=polyder(p,q)  
[k,d]=polyder(p,q)
```

```
I=polyint(p,c)  
I=polyint(p)
```

```
y=polyval(p,x)  
Y=polyvalm(p,A)
```

```
x=roots(p)
```

```
poly2sym(p), sym2poly(f), poly(x)
```

多项式运算中，
使用的是多项式
系数向量，
不涉及符号计算！

内容提要

■ 多项式运算

- 多项式转化为符号表达式: `poly2sym`, `sym2poly`
- 四则运算: `conv`、`deconv`
- 导数与积分: `polyder`、`polyint`
- 求值与零点: `polyval`、`polyvalm`、`roots`、`poly`

■ 代数方程求解

- 线性方程组数值求解: `linsolve`
- 非线性方程数值求解: `fzero`
- 非线性方程符号求解: `solve`

线性方程组求解

- 线性方程组求解: `linsolve`

```
x=linsolve(A,b)
```

解线性方程组 $Ax = b$

例：解方程组

$$\begin{cases} x + 2y - z = 2 \\ x + z = 3 \\ x + 3y = 8 \end{cases}$$

```
A=[1 2 -1; 1 0 1; 1 3 0];
```

```
b=[2; 3; 8];
```

```
x=linsolve(A,b)
```

- 注意：这里的右端项 b 必须是列向量！

非线性方程求解

● 非线性方程求解：`fzero`

<code>x=fzero(f,x0)</code>	求方程 $f(x)=0$ 在 x_0 附近的一个解
<code>x=fzero(f,[a,b])</code>	找出区间 $[a,b]$ 内的一个解

- 方程可能有多个根，但 `fzero` 只给出 x_0 附近的一个
- x_0 是一个标量，为参考点，不能缺省
- `fzero` 先找出一个包含 x_0 的区间，使得 f 在这个区间的两个端点上的函数值异号，然后再在这个区间内寻找方程 $f(x)=0$ 的解；如果找不到这样的区间，则返回 `NaN`
- 由于 `fzero` 是根据函数是否穿越横轴来决定零点，因此它无法确定函数曲线仅触及横轴但不穿越的零点，如 $|\sin(x)|$
- $f(x)=0$ 在 $[a,b]$ 内可能有多个解，但 `fzero` 只给出一个

非线性方程求解

- 非线性方程求解：`fzero`

<code>x=fzero(f,x0)</code>	求方程 $f(x)=0$ 在 x_0 附近的一个解
<code>x=fzero(f,[a,b])</code>	找出区间 $[a,b]$ 内的一个解

- 参数 `f` 是一个函数句柄，可通过以下方式给出
 - 匿名函数，如：`f=@(x) x^3-3*x+1`
 - 函数文件（略）

注意：f 不是方程！也不能使用符号表达式！

fzero 举例

例：求 $f(x)=x^3-3x+1$ 在区间 $[-2, 0]$ 内的实根。

```
f=@(x) x^3-3*x+1;  
x0=fzero(f, [-2,0])
```

例：求 $f(x)=\sin(x)$ 在 10 附近的实根。

```
fzero(@sin, 10)
```

用 **fzero** 求零点时可以先通过作图确定零点的大致范围

代数方程符号求解

- 代数方程符号求解: `solve`

<code>s=solve(f,v)</code>	关于指定自变量求解
<code>s=solve(f)</code>	关于默认变量求解

- 参数 `f` 是一个符号表达式
- 也可以是用等号 `==` 连接的方程 (Matlab新版本)

例: 解方程 $x^3 - 3x + 1 = 0$

```
syms x; f=x^3-3*x+1;  
sol=solve(f,x) % 或 s=solve(f)
```

```
syms x;  
sol=solve(x^3-3*x+1==0,x) % MatlabR2012版本
```

代数方程符号求解

- `solve` 也可以用来解方程组

```
solve(f1, f2, ..., fN, v1, v2, ..., vN)
```

- 求由 f_1, f_2, \dots, f_N 确定的方程组关于 v_1, v_2, \dots, v_N 的解

例：解方程组

$$\begin{cases} x + 2y - z = 27 \\ x + z = 3 \\ x^2 + 3y^2 = 28 \end{cases}$$

```
syms x y z;  
[s1,s2,s3]= ...  
    solve(x+2*y-z-27, x+z-3, x^2+3*y^2-28, x,y,z)
```

- 注意输出变量的对应顺序！
- 如果得不到解析解，`solve` 会给出数值解

求解方程函数小结

roots(p): 多项式的所有零点, **p** 是多项式系数向量

fzero(f, x0): 求 $f(x)=0$ 在 **x0** 附近的一个根, **f** 是函数句柄, 可以通过匿名函数或函数文件来定义, 但不能是方程或符号表达式!

linsolve(A, b): 解线性方程组

solve(f, v): 求方程关于指定自变量的解, **f** 是符号表达式或符号方程;

- **solve** 也可解方程组 (包含非线性)
- 得不到解析解时, 给出数值解

上机作业

1、已知多项式 $p(x) = 6x^4 - 5x^2 + x$, $q(x) = 6x^3 - 1$

计算 $p(x) + q(x)$, $p(x)q(x)$, $\frac{p(x)}{q(x)}$ 及它们的导数

(将所用命令写入文件 **m61.m**)

2、已知多项式 $p(x) = 816x^3 - 3835x^2 + 6000x - 3125$

(a) 求出 $p(x)$ 的所有零点;

(b) 用 **fzero** 计算 $p(x)$ 的第二大零点

(将所用命令写入文件 **m62.m**)

3、求方程组 $\begin{cases} x^2 + y^2 = 4 \\ x^2 - y = 1 \end{cases}$ 的解

(将所用命令写入文件 **m63.m**)

上机作业

4、已知 Chebyshev 多项式定义如下：

$$T_0(x) = 1, \quad T_1(x) = x,$$

$$T_{n+1}(x) = 2x T_n(x) - T_{n-1}(x), \quad n = 1, 2, \dots$$

试编程计算 $T_{20}(x)$ 的导数（注： $T_n(x)$ 为 n 阶多项式）

(将程序取名为 `m64.m`)

(注：要求利用多项式运算，不要使用符号计算！)

5、编写一个函数文件：`m65.m`，实现两个向量的加运算
(在长度较短的向量前面添 0，使得两个向量长度相等)

上机要求

□ 上机要求

- 将所有文件作为附件，通过 foxmail 以邮件形式发给 `mhjs@system.mail`
- 邮件主题为：**机号-学号-姓名**，其中机号为 **两位数**
- 三个字段之间用英文状态下的减号连接
- 每个 M 文件的第一行添加一条注解语句：
% 机号-学号-姓名