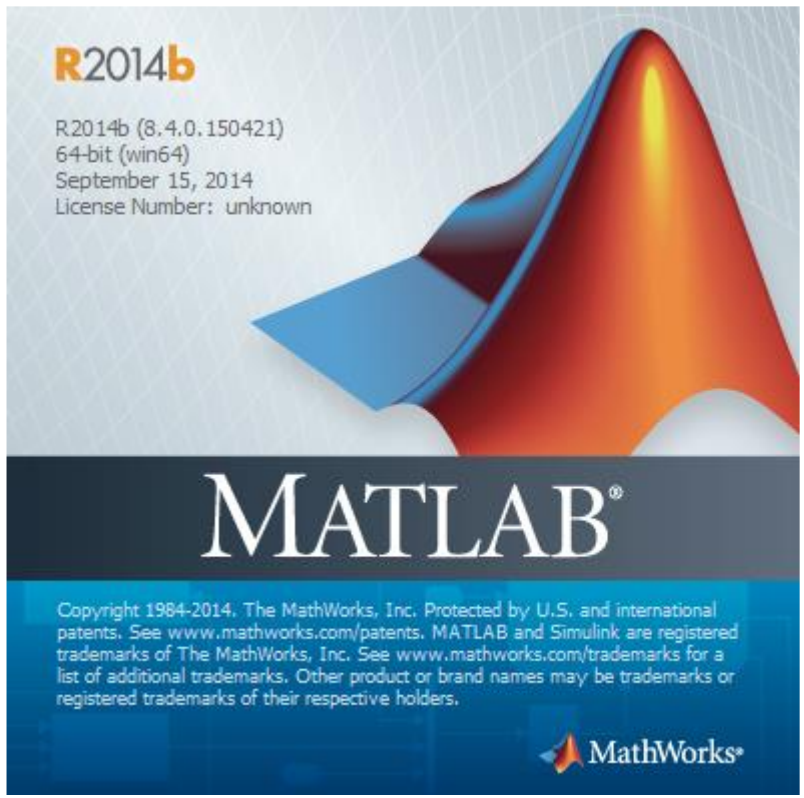
第5章

MATLAB软件与应用



第五章 数值计算

- 线性方程组
- 多项式与非线性方程组
- 插值与拟合
- 数值积分
- 补充

本课程不过多关注数学，而是强调充分利用matlab的现有功能与函数，更简便快捷的完成数学方面的工作。

数值计算的意义

数值计算(近似)　VS　解析计算(精确)

许多科学和工程问题可归结为下列数学问题

方程求根　　求解线性/非线性方程组

定积分　　求解常/偏微分方程

对他们进行数值计算而不是解析计算的原因:

1. 解析解或解析表达式存在，但很难处理或计算量过大
2. 解析解不存在（或无法证明其存在性）
3. 问题本身很难建立或根本没有解析表达式
4. 工程实践中允许合理误差的存在

数值计算的意义

数值计算放弃精确解析解，更容易地获取近似解

- 高次(>=5次)代数方程，例如 $x^5 - 3x + 1 = 0$
- 超越方程，例如 $e^{-x} – \cos(x) = 0$
- 看似简单却极难进行解析求解。

$\int_a^b \frac{\sin x}{x} dx$ 无法用初等函数进行解析表示。

n阶线性方程组可用克莱默法则理论上可以得到解析解，但算法复杂度高达n!(n-1)(n+1)，计算量惊人！例如n=30，大约需10^{35}次乘法运算，即使采用世界TOP1“太湖之光”超级计算机（10^{17} 次乘法/秒），

也需要数十亿年！！！

解线性方程组

数值计算 解线性方程组

线性方程组： $Ax = b$

A是已知m×n阶矩阵；
b是已知m阶列向量；
x是待求的n阶列向量；
Ax-b为残值向量，常用R表示。

线性方程组求解，就是找到使R尽量小的x，
理想条件是R为零向量，其模等于0。

数值计算 解线性方程组

定解

方程和未知数数量相等，且系数矩阵A线性无关

奇异

系数矩阵A线性相关

A = [2,3; -4,-6];

b = [1; -2];

x = A\b

不定

方程数量少于未知数无法求解或只能得一个特解

超定

方程数量多于未知数一般可得最小二乘解（残值模尽量小，但不为**0**）

数值计算 解线性方程组 A\b

定解

反除和求逆解法（用于定解型）

对于 Ax = b

x=A\b 或 x=inv(A)*b

以上两个式子从数学上等价，但由于matlab内部处理方式的区别，反除运算速度更快，不推荐用inv(A)来求解方程组。

```
A(1,:)=[5,4,-4]; A(2,:)=[2,-5,6]; A(3,:)=[-1,2,-3];
b=[11,15,-9]'; x=A\b
```

结果是 x =

3.0000
3.0000
4.0000

数值计算 解线性方程组 A\b

超定

方程数量多于未知数
一般可得最小二乘解

多项式拟合一般可归结为超定线性方程组

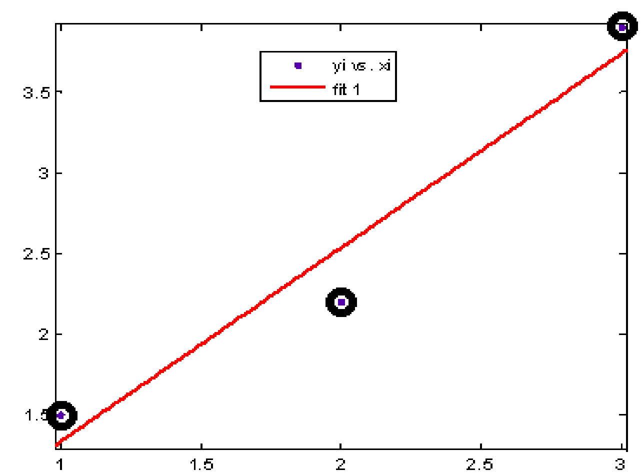
【例】用线性多项式(2个待定系数)来拟合3个数据点。

xi=[1, 2, 3], yi=[1.5, 2.2, 3.9],

可归结为超定线性方程组

$$\begin{aligned} y(p_1,p_2,xi_1)&=y_1 \\ y(p_1,p_2,xi_2)&=y_2 \\ y(p_1,p_2,xi_3)&=y_3 \end{aligned} \quad\Rightarrow\quad \begin{aligned} p_1+p_2&=1.5 \\ 2p_1+p_2&=2.2 \\ 3p_1+p_2&=3.9 \end{aligned}$$

```
xi=[1, 2, 3],  yi=[1.5, 2.2, 3.9],
A=[xi;1,1,1]', b=yi'
x=A\b %所得x不能使残值归零
```



数值计算 解线性方程组 A\b

不定

A(1,:)=[1,1,1]; A(2,:)=[1,-1,1];
b=[0,0]'; **x=A\b**

结果是 x =

0
0
0

以上为不定线性方程组，方程数量少于未知数，有多解(例如 [1,0,-1]是另一个解)，A\b只给出一个特解。

数值计算 解线性方程组 A\b

计算机只保留有限个有效数字，
额外部分将被舍掉，形成舍入误差

线性方程组的病态问题

有些线性方程组虽然可解，但由于数值计算舍入误差的影响而难以得到足够精确的解，这类方程组的系数矩阵A一般具有很大的条件数 cond(A)。

- 有的数值算法将舍入误差急剧放大，而使最终结果的误差难以接受，甚至根本无法完成求解。
- 更好的算法对舍入误差不敏感，从而改善病态问题。
- 还有所谓的预处理方法，可在求解前对A矩阵进行加工，削弱其病态。
- **A\b**的核心算法没有公开，但根据其表现可能是高斯消元及其变种，对高维问题计算量仍然惊人，对病态问题也只能碰运气。

数值计算 解线性方程组 A\b

【例】A是希尔伯特矩阵 A(i,j)=1/(i+j-1)，

x=[1;1;1;....]，b=A*x，构造方程组 A x = b，其精确解为[1;1;1...]。对于4阶希尔伯特矩阵

$$A=\begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix}$$

cond(A)=1.55x10^4,条件数不大,
A\b能较精确的得到该解。

而对于13阶希尔伯特，条件数高达10^{17}，
A\b结果是

1.000000079088454
0.999987067979130
1.000513120201064
0.99127267554746
1.079665518533795
0.562206815473678
2.543650222515574
-2.611079616472801
6.665505232951684
-4.893541759955957
4.898856297509340
-0.484840311408609
1.247804755845142

如何解决？？

数值计算 解线性方程组

方法汇总

1. 直接解法

- **反除**（本质上很可能仍是高斯消元类算法）
- **高斯消元**（及列主元消元、高斯-约当消元等变种）
- **LU分解**（计算能力整体上同高斯消元基本相当）
- `linsolve(A,b)`（核心仍然是LU分解）

2. 迭代解法

- 雅可比迭代、高斯-赛德尔迭代、超松弛法等

数值计算 解线性方程组 更好的算法

3. 更好的迭代解法

CG	共轭梯度算法
PCG	预处理共轭梯度算法
BICGSTAB	双共轭稳定算法
GMRES	广义残值最小算法

....

这类算法自编程序并不复杂，核心代码只有十几行，可参考数值分析教材，或直接调用matlab集成的 **cgs，pcg，bicg，gmres 等。** help pcg 可查看更多类似函数。

对于高维病态问题，例如13阶以上希尔伯特方程组，反除、高斯消元、LU分解基本无效，高斯-赛德尔等也需要数十万次迭代，才能得到比较好的结果，而CG等算法一般只需十余次迭代，效率很高。

没有万能算法！

多项式与非线性方程（组）

数值计算　多项式

多项式　$p(x)=a_0x^n+a_1x^{n-1}+\cdots a_{n-1}x+a_n$

可表示为行向量　$p=[a_0,a_1,\cdots,a_{n-1},a_n]$

向量元素按降幂顺序用作多项式系数值。

多项式操作常用函数

函数名	说明	函数名	说明
roots	多项式求根	**polyfit**	多项式拟合
poly	由根创建多项式	**polyder**	求多项式导数
polyval	多项式求值	**conv**	多项式乘法
polyvalm	矩阵多项式求值	**deconv**	多项式除法
poly2sym	多项式转为 符号表达式	…	…

数值计算　多项式

[例] 多项式可以用系数向量来表示，也可以转化为符号多项式。

```
p=[1  -5   6  -33];
s=poly2sym(p)  %将多项式向量表示成符号多项式
        s =
              x^3-5*x^2+6*x-33
```

数值计算　多项式

roots 求多项式方程在复数范围内的根。

poly roots的逆运算，由根来创建多项式。

```
p=[2 6 4]
poly2sym(p)
r=roots(p)
p1=poly(r)
```

结果是

```
p=
        2   6   4
ans =
        2*x^2+6*x+4
r =
        -2
        -1
p1 =
        1   3   2
```

注意：**poly**创建的多项式最高次项的系数自动归一化为**1**

数值计算　多项式

polyval(p,v)

计算多项式**p**在**v**处的值，**v**是矩阵时进行点运算。

polyvalm(p,v)

计算多项式**p**在**v**处的值，**v**是矩阵，进行矩阵幂运算。

```
p=[1, 2, 3];   b=[1  1;1  1];  polyval(p,b)
 ans = 6   6
       6   6
polyvalm(p,b)
 ans =   7    4
         4    7
x^2+2*x+3*eye(2)
 ans =   7    4
         4    7
```

此两段等价

数值计算补充　多项式

多项式的乘法函数**conv**，等同于向量卷积； **what ?**

多项式的除法函数**deconv**，等同于向量解卷。

```
p=[1  2  3]; poly2sym(p)
        ans =
                x^2+2x+3

d=[2   5]; poly2sym(d)
        ans =
                2x+5
pd=conv(p,d); poly2sym(pd)

        ans =
                2*x^3+9*x^2+16*x+15

p1=deconv(pd,d)   %显然与p相同
        p1 =
           1    2     3
```

数值计算补充　多项式

polyder 多项式的微分

```
p=[2 -5 6 -1 9]; poly2sym(p)
        ans =
        2*x^4-5*x^3+6*x^2-x+9

dp=polyder(p)
        dp =
        8   -15    12    -1

poly2sym(dp)
        ans =
        8*x^3-15*x^2+12*x-1
```

数值计算 非线性方程(组)

对于一般非线性方程(组)

$f_1(x)=0, \cdots, f_n(x)=0, \quad x=(x_1, \cdots, x_n)$

最常用的求解命令是 **solve** 和 **fsolve**

solve (‘f1(x)’,…,‘fn(x)’)

实质上是符号运算

fsolve (‘f1(x)’,…,‘fn(x)’,**初值**,option)

是基于迭代算法的数值运算

数值计算 非线性方程(组) solve

solve并非万能，对于单方程，一般可求解，
对于复杂方程(组)，有时会失效。

【例】 解方程 $x^3 = -x-2$

```
s=solve('x^3=-x-2')
```

或

```
s=solve('x^3+x+2')
double(s) %这样能得到数值
```

```
s=
        -1
        (7^(1/2)*i)/2 + 1/2
        1/2 - (7^(1/2)*i)/2
ana =
        -1.0000 + 0.0000i
        0.5000 + 1.3229i
        0.5000 - 1.3229i
```

【例】 解方程 $ax^2+bx+c = 0$

```
solve('a*x^2+b*x+c')
ans=
-1/2*(b-(b^2-4*a*c)^(1/2))/a
-1/2*(b+(b^2-4*a*c)^(1/2))/a
```

非线性方程(组)中
多项式方程是比较容易求解的。
其他形式求解难度更高，solve
有时给出错误结果或不能求解。

原因：符号运算本质上是在
推导解析公式，难度很高。

数值计算 非线性方程(组) solve

【例】 $x^2 y^2 = 0$

$$x - \frac{y}{2} = c$$

[x1,x2]=solve('x^2*y^2,x-(y/2)-c') %默认未知数x y

结果是 x1 = c 对应公式中x

0

x2 = 0 对应公式中y

-2*c

[x1,x2]=solve('x^2*y^2,x-(y/2)-c', 'x', 'c') %指定未知数x c

结果是 x1 = 0 对应公式中x

x2 = -y/2 对应公式中c

数值计算 非线性方程(组) solve

【例】 $$\begin{cases} \sin x + y^2 + \ln z - 7 = 0 \\ 3x + 2^y - z^3 + 1 = 0 \\ x + y + z - 5 = 0 \end{cases}$$

```
[x,y,z]=solve('sin(x)+y^2+log(z)-7=0',...
              '3*x+2^y-z^3+1=0',...
              'x+y+z-5=0', 'x','y','z')
```

结果：

x = 5.1004127298867761621009050441017

y = -2.6442371270278301895646143811868

z = 2.5438243971410540274637093370850

数值计算 非线性方程(组) fsolve

fsolve 本质上是基于迭代算法的数值求解

方程组需要通过M函数文件来定义

x = fsolve (‘fun’, X0, options)

M函数文件fun.m

function f = fun(x)

f(1)= $f_1(x)$ **;**

……

f(n)= $f_n(x)$

初值

options=1表示输出中间结果，可省略

数值计算 非线性方程(组) fsolve

【例】函数文件**myeq001.m**

```
function eq=myeq001(f)
x=f(1);y=f(2);
eq(1)=x^3-y^2;
eq(2)=exp(-x)-y;
```

在命令行运行

y=fsolve('myeq001',[1,1],1)

结果：**y = 0.6488   0.5226**

【例】对于简单方程，也可用内联函数或匿名函数。

```
fun1=inline('sin(2*x+1)')
fun2=@(x)sin(2*x+1)
y1=fsolve(fun1,[-9 9])  %分别给两个初值，得到两个解
y2=fsolve(fun2,[-9 9])
```

数值计算 非线性方程(组) roots

roots 仅用于多项式方程求根。

【解】计算多项式方程 $2x^5+17x^4-x^3-157x^2-x=-140$ 的根。

p=[2,17,-1,-157,-1,140]; root(p)

结果为

```
-7.000
-4.000
       2.500
      -1.000
       1.000
```

高于5次的多项式方程用roots可能无法求解。

插值与拟合

数值计算 一维插值

yi=interp1(x，y，xi，'method', option)

- yi：xi处的插值结果
- x，y：插值节点
- xi：待求点
- 'method'：插值方法

如果x缺省，则x用数组下标1~n代替，n为y向量的长度。要求x单调，一般xi不能够超过x的范围，否则要在method之后加上’extrap’来表示可进行外推插值。

Method可取：

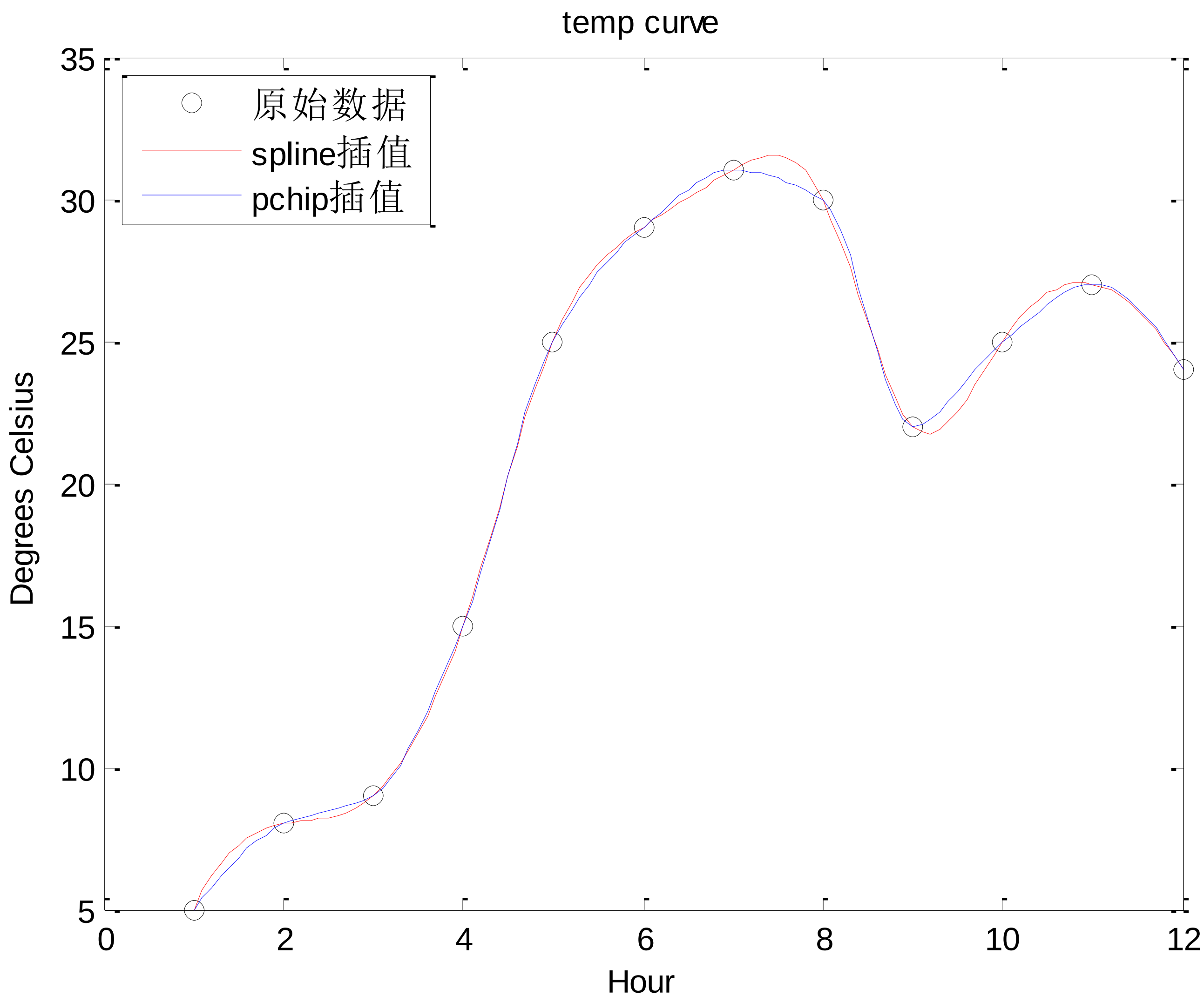
nearest ：最邻近插值
spline ：三次样条插值
cubic ：三次多项式插值
pchip ：三次赫尔米特插值
linear ：分段线性插值（默认）

数值计算 一维插值

【例】在1-12的11小时内，每隔1小时测量一次温度，依次为：5，8，9，15，25，29，31，30，22，25，27，24。试估计每隔0.1小时的温度并绘图。

```
hours=1:12;
temps=[5 8 9 15 25 29 31 30 22 25 27 24];
h=1:0.1:12;
t1=interp1(hours,temps,h,'spline');
t2=interp1(hours,temps,h,'pchip');
plot(hours,temps,'ko',h,t1,'r-',h,t2,'b-')
title('temp curve'),
xlabel('Hour'),ylabel('Degrees Celsius')
legend('原始数据','spline插值','pchip插值','Location','NorthWest')
```

数值计算 一维插值


temp curve
原始数据
spline插值
pchip插值
35
30
25
20
15
10
5
Degrees Celsius
0
2
4
6
8
10
12
Hour

数值计算 二维插值

第一类 网格型二维插值

已知 $m \times n$个节点

(x_i, x_j, z_{ij}) $(i=1, 2, \ldots, m;\ j=1, 2, \ldots, n)$

其中x_i, y_j互不相同，设

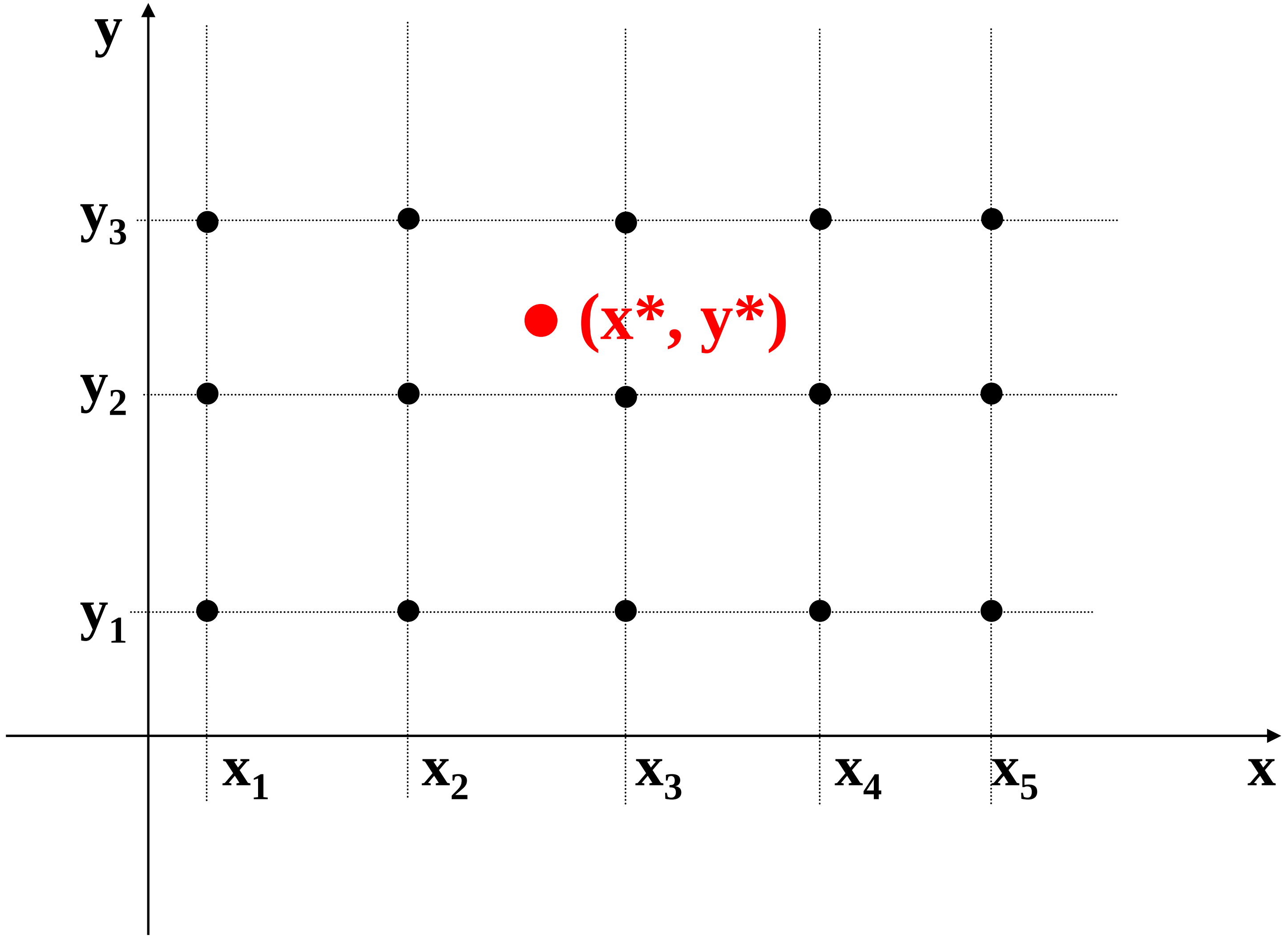
$$xmin=x_1<x_2<\ldots<x_m=xmax$$

$$ymin=y_1<y_2<\ldots<y_n=ymax$$

求任一插值点(x^*, y^*) 处的插值Z^*

数值计算 二维插值

第一类 网格型二维插值



数值计算 二维插值

interp2 网格型二维插值，二维版的interp1

```
z=interp2(x0,y0,z0,x,y,'method')
```

被插值点的函数值

插值节点

被插值点

插值方法

Method可取：

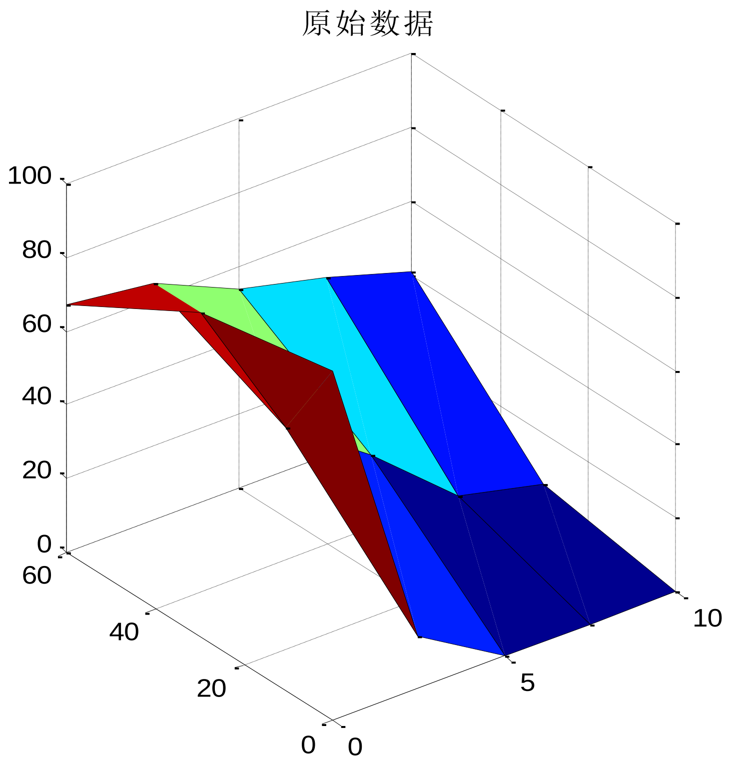
nearest	最邻近插值
linear	双线性插值
cubic	双三次插值
spline	双元样条
缺省时	双线性插值

数值计算 二维插值

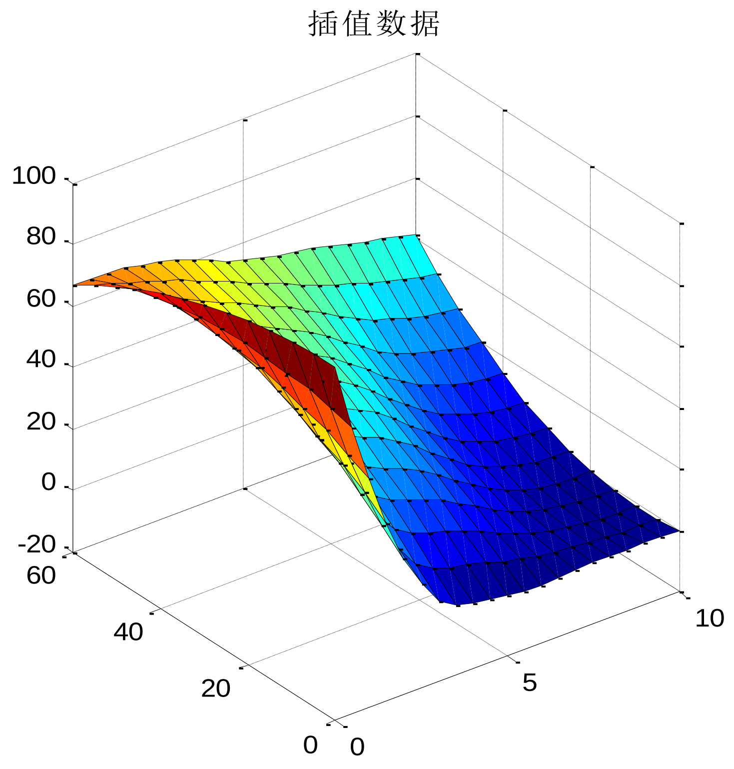
【例】对一根10米长钢轨进行温度传播测试。x表示测量点0:2.5:10(米)，h表示测量时间0:30:60(秒)，T表示得的温度(℃)。试用插值求出在一分钟内每隔5秒、钢轨每隔0.5米处的温度TI。

```
[x,h]=meshgrid(0:2.5:10,0:30:60)
T=[95,14,0,0,0;88,48,32,12,6;67,64,54,48,41];
[xi,hi]=meshgrid(0:0.5:10,0:5:60)
TI=interp2(x,h,T,xi,hi,'cubic');
subplot(1,2,1)
surf(x,h,T)  ,title('原始数据');
subplot(1,2,2)
surf(xi,hi,TI),title('插值数据');
```

数值计算 二维插值



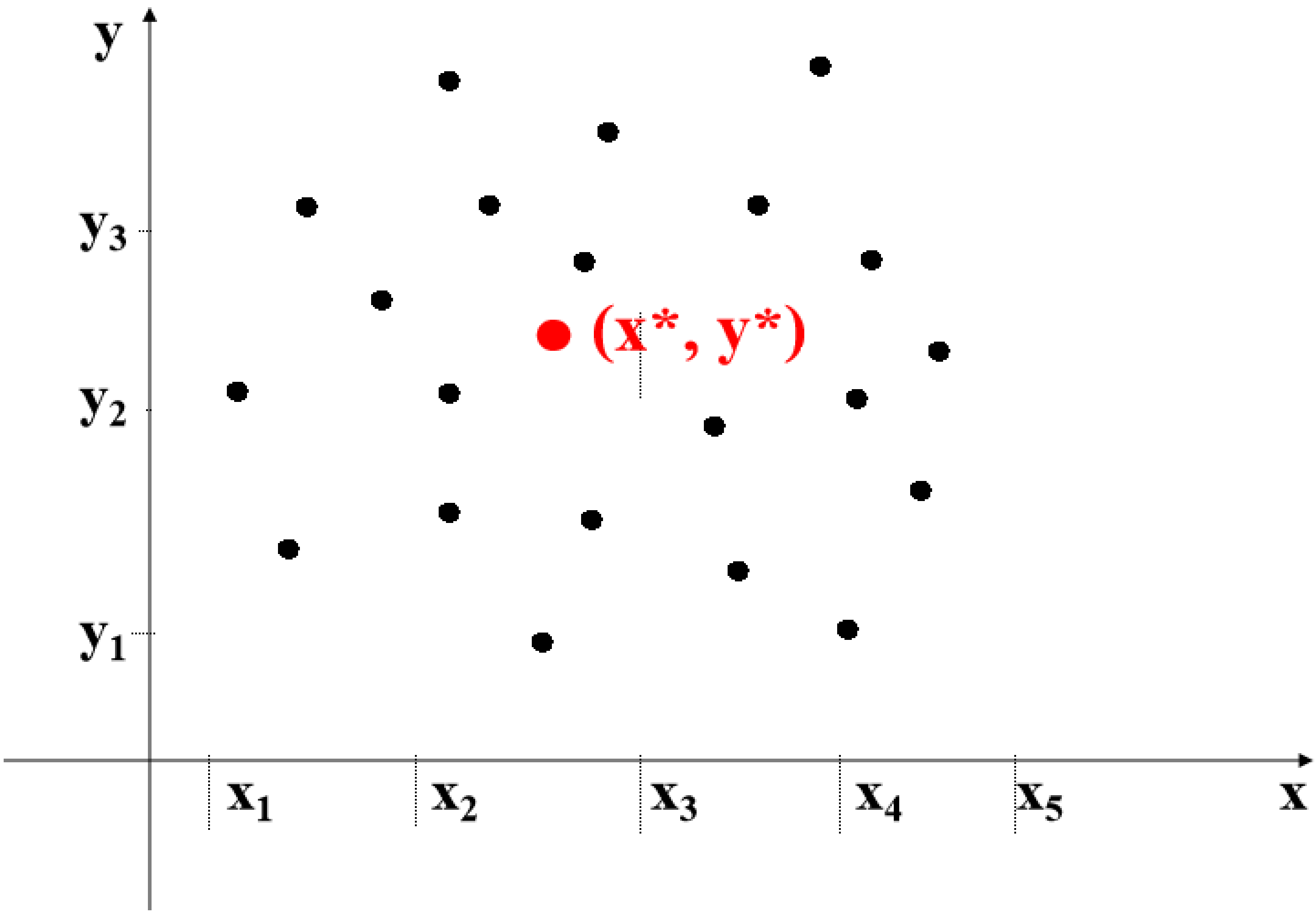
原始数据
100
80
60
40
20
0
60
40
20
0
0
5
10



插值数据
100
80
60
40
20
0
-20
60
40
20
0
0
5
10

数值计算 二维插值

第二类 散乱点型二维插值（包含第一类）



数值计算 二维插值

griddata 散乱点型二维插值

```
z=griddata(x0,y0,z0,x,y,'method')
```

被插值点的函数值

插值节点

被插值点

Method可取

nearest， natural，cubic

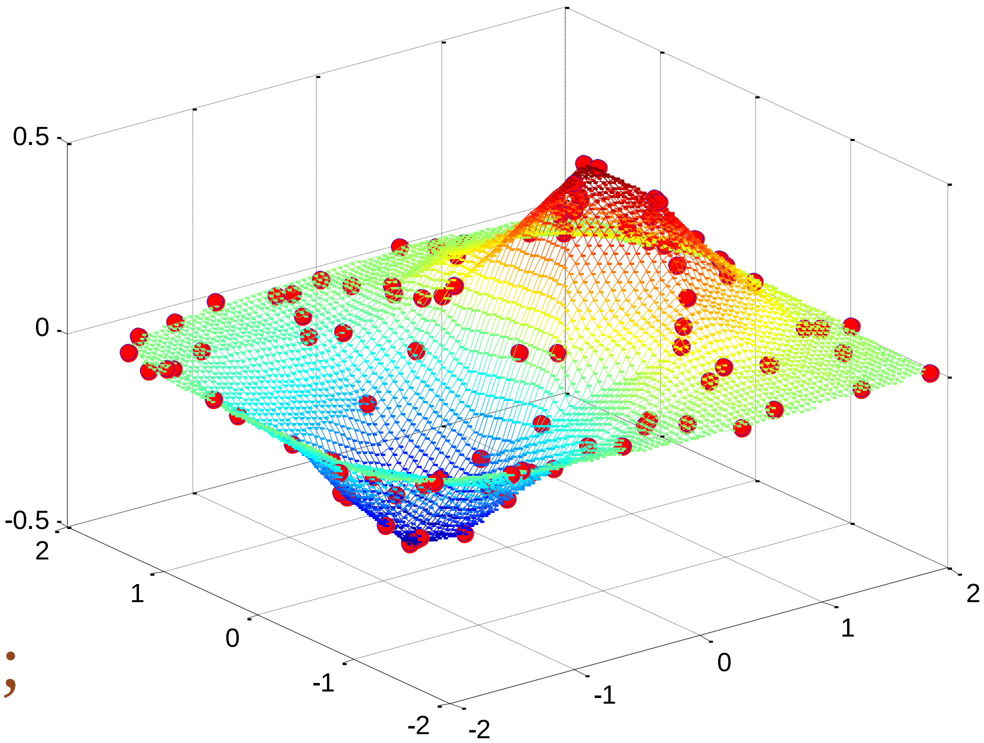
linear（缺省值）

v4（4格点样条函数）

数值计算 二维插值

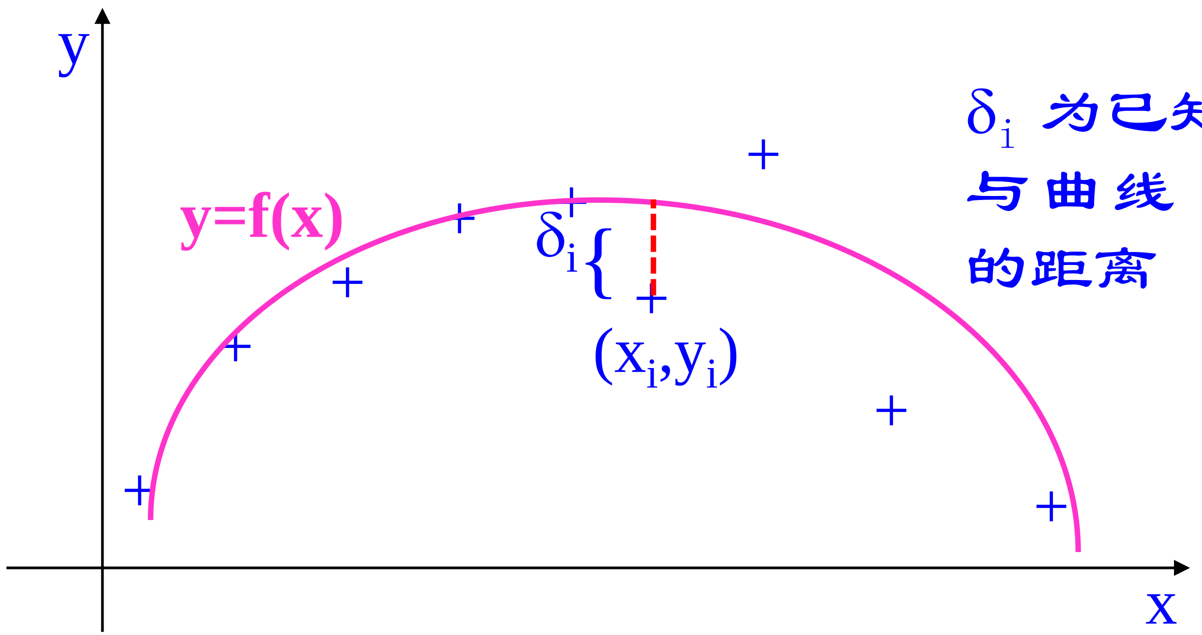
griddata 散乱点型二维插值

```
rand('seed',0)
x = rand(100,1)*4-2;
y = rand(100,1)*4-2;
z = x.*exp(-x.^2-y.^2);
ti = -2:0.05:2;
[XI,YI] = meshgrid(ti,ti);
ZI = griddata(x,y,z,XI,YI);
mesh(XI,YI,ZI), hold
plot3(x,y,z,'o', 'MarkerFaceColor',[1,0,0]);
hidden off
```



数值计算 曲线拟合

曲线拟合：由已知离散点构造函数(最典型的是多项式函数)，使函数曲线在某种准则下最接近已知点（曲线不要求通过已知点，如果通过则成为插值）



δ_i 为已知节点 (x_i, y_i) 与曲线 $y=f(x)$ 之间的距离

为什么不直接用插值代替拟合?

插值函数通过已知节点，是否比拟合更精确?

数值计算 曲线拟合

最小二乘曲线拟合的基本原理

步骤：1）选定一类函数（可以是多项式，三角函数等）

$f(x，a_1, a_2, \cdots, a_m)$

其中 $a_1, a_2, \cdots a_m$ 为待定常数。

2）确定参数 $a_1, a_2, \cdots a_m$

最小二乘准则：使n个已知点（x_i, y_i）与曲线 $y=f(x，a_1, a_2, \cdots, a_m)$ 的距离 δ_i 的平方和最小。

数值计算 曲线拟合

已知点数量多于待定系数，产生超定线性方程组，一般采用优化方法来解

数学模型

$$J(a_1,a_2,\cdots a_m)=\sum_{i=1}^{n}\delta_i^2$$

$$=\sum_{i=1}^{n}[f(x_i,a_1,\cdots,a_m)-y_i]^2$$

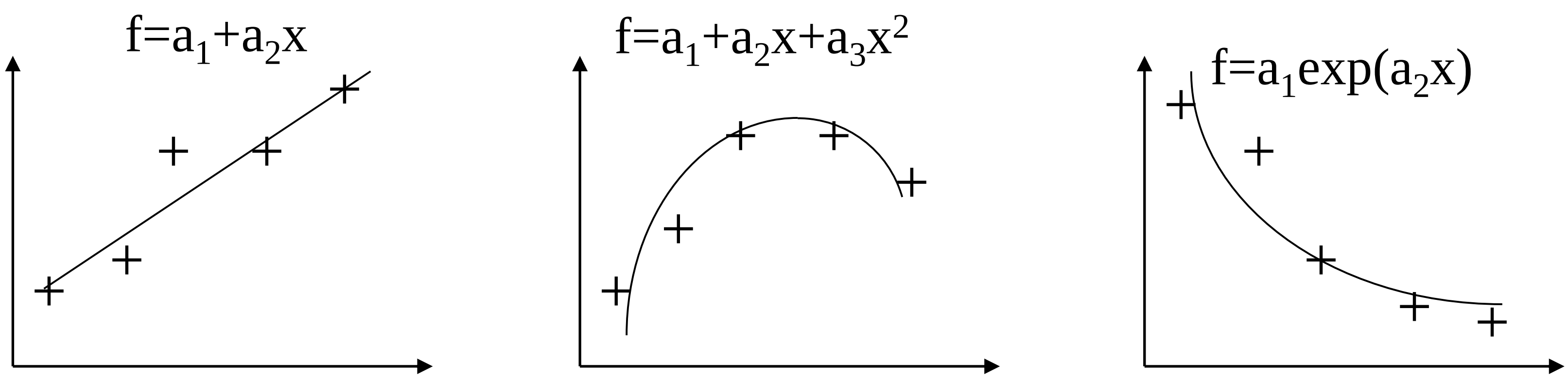
问题归结为:

求 a_1, a_2, $\cdots a_m$ 使 $J(a_1, a_2, \cdots a_m)$ 最小。

数值计算 曲线拟合

最小二乘拟合函数类型的选取（需经验或多次尝试）

1. 将数据作图，确定 f 的类型：



2. 通过机理分析建立数学模型来确定

例如：炮弹发射后的轨迹中得到的炮弹位置离散数据，可以二阶多项式（抛物线）

数值计算 曲线拟合

1. 多项式拟合（polyfit）

```
a=polyfit(x,y,m)
```

- a：系数
- x,y：数据点
- m：拟合多项式次数

polyfit可做拟合和插值，具体情况根据设定的多项式次数和数据点数量来确定。

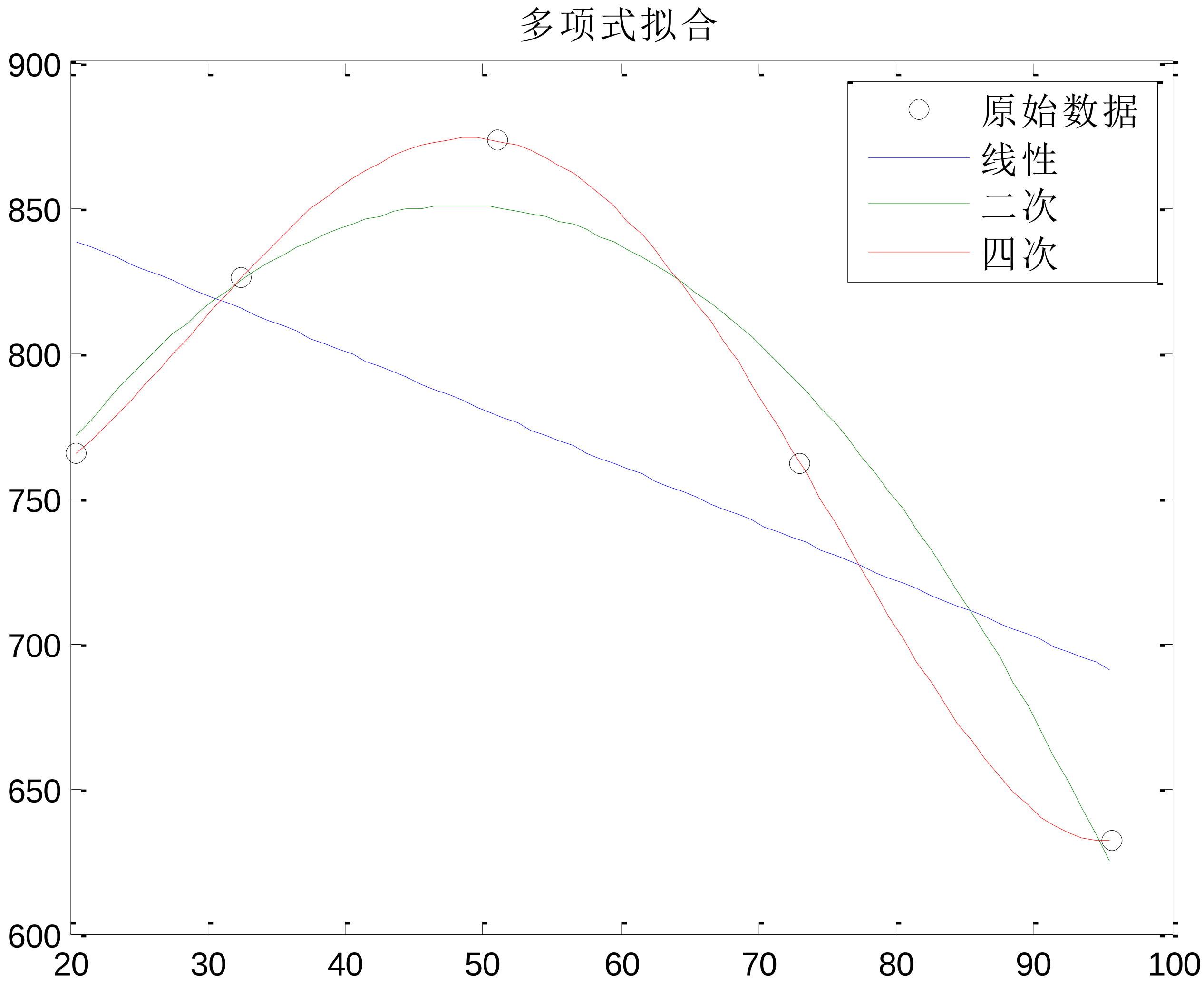
例如5数据点可做4次多项式插值，或<=3次拟合，而4次以上则无法完成拟合或插值。

数值计算 曲线拟合

例：有5个反映参数 r 随着 t 变化的数据点，求拟合多项式。

```
t=[20.5 32.5 51 73 95.7]; %5个数据点，4次拟合实质上是插值
r=[765 826 873 762 632];
tnew=20.5:95.7;
aa1=polyfit(t,r,1); aa2=polyfit(t,r,2); aa4=polyfit(t,r,4);
a1=aa1(1); a2=aa2(1); a4=aa4(1);
b1=aa1(2); b2=aa2(2); b4=aa4(2);
y1=polyval(aa1,tnew); y2=polyval(aa2,tnew);
y4=polyval(aa4,tnew);
plot(t,r,'ko',tnew,y1,tnew,y2,tnew,y4)
title('多项式拟合');
legend('原始数据','线性','二次','四次')
```

数值计算 曲线拟合



多项式拟合
900
850
800
750
700
650
600
20
30
40
50
60
70
80
90
100
原始数据
线性
二次
四次

数值计算 曲线拟合

2. 一般的最小二乘曲线拟合lsqcurvefit

```
[a,resnorm,residual]=lsqcurvefit(f,a0,x,y)
```

a：待定常数

resnorm：误差平方和

residual：各数据点处的误差

f：函数M文件，作为拟合函数

a0：a 的初值

x, y：数据点

拟合函数类型由M函数文件指定

数值计算 曲线拟合

【例】用函数

$$y(x)=a_1*\exp(-a_2*x)+a_3*\exp(-a_4*x)$$

拟合下列数据点：

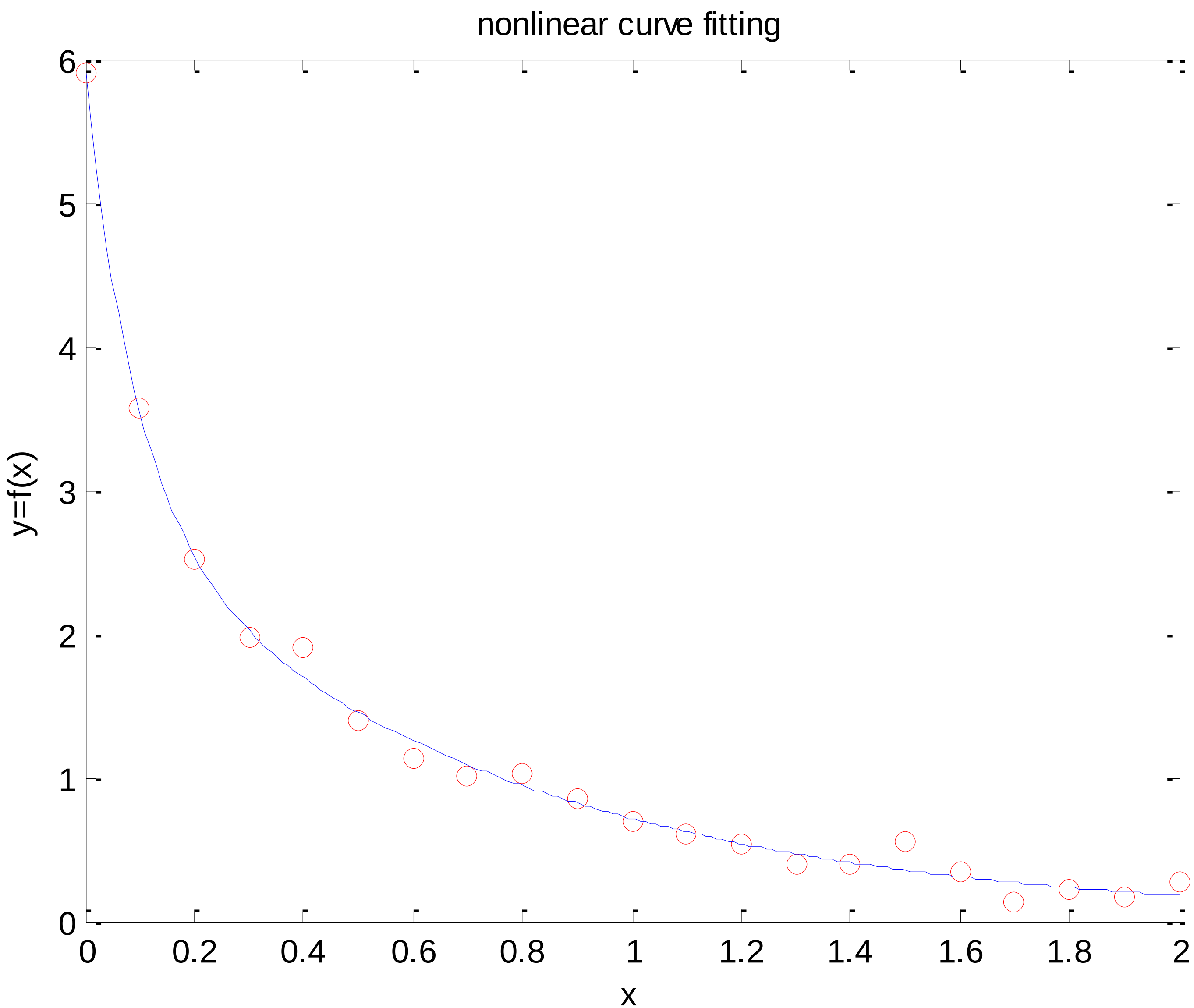
```
xdata=[0:0.1:2];
ydata=[5.8955   3.5639   2.5173   1.9790   1.8990 ...
    1.3938   1.1359   1.0096   1.0343   0.8435   0.6856 ...
    0.6100   0.5392   0.3946   0.3903   0.5474   0.3459 ...
    0.1370   0.2211   0.1704   0.2636];
```

数值计算 曲线拟合

```
function y=myfitfun1(a,x )      %保存为myfitfun1.m
y=a(1)*exp(-a(2)*x)+a(3)*exp(-a(4)*x);

%命令行运行
a0=[1,1,1,0];
[a,resnorm,residual,flag,output]=lsqcurvefit(‘myfitfun1',a0,xdata,ydata);
xi=linspace(0,2,200);
yi=myfitfun1(a,xi);
plot(xdata,ydata,'ro',xi,yi)
xlabel('x'),ylabel('y=f(x)'),
title('nonlinear curve fitting')
```

数值计算 曲线拟合

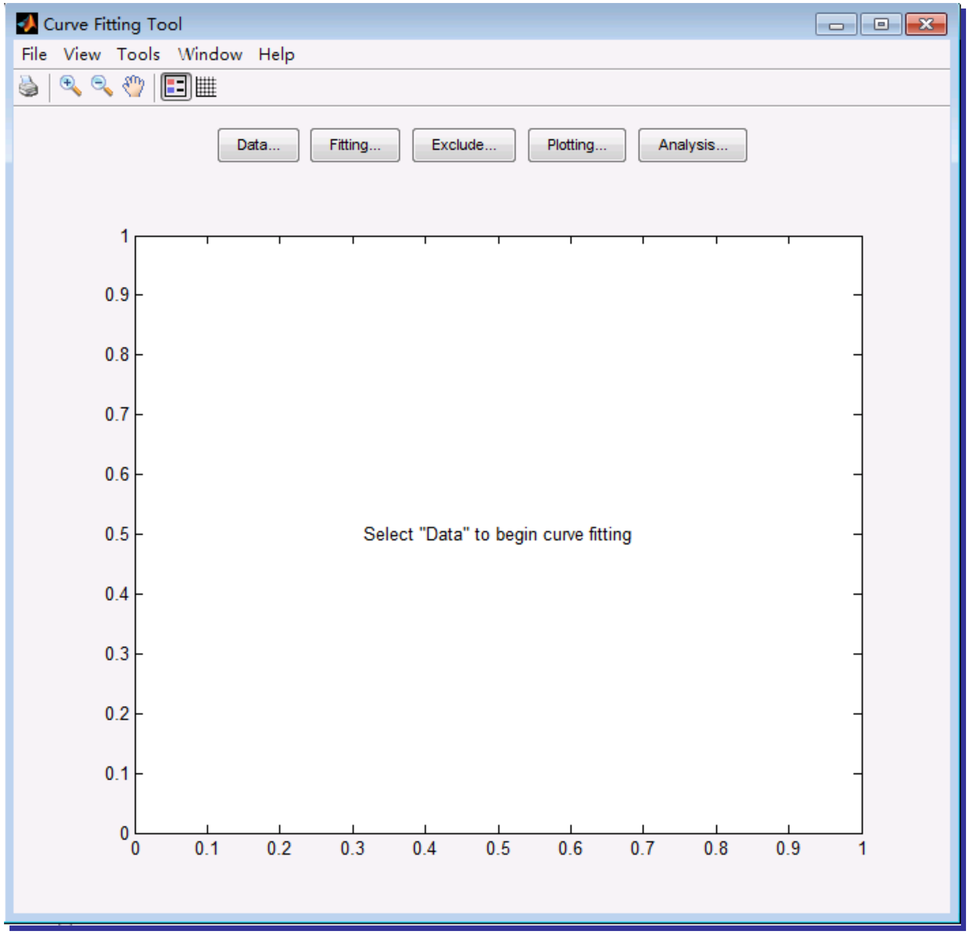


nonlinear curve fitting
y=f(x)
x
0
1
2
3
4
5
6
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

数值计算 曲线拟合

3. curve fitting工具箱

一种基于交互式界面的插值与拟合软件，可以在命令窗口输入cftool来调用。该工具箱的功能，基本都可以通过编程来实现。在工具箱中进行的操作，还能由系统转换为程序代码。



数值积分

数值计算 数值积分的必要性

- 原函数计算的困难性

利用原函数计算定积分的方法基于牛顿-莱布尼兹公式。但多数情况下，原函数无法用初等函数表示，或没有解析解。

- 实际应用的限制

对很多实际应用，只能知道被积函数的离散数值，或被积函数是微分方程的解，或积分区域是复杂高维空间，就只能使用数值积分来进行近似计算。

数值计算 数值积分

功能有限
适用范围窄

广泛应用

数值积分

符号积分

数值积分基本原理

将积分区域分成n个子区间（少数情况也可不分），利用**插值或拟合**技术在每个区间上分别构造近似函数，对各近似函数进行定积分解析计算，再进行求和。

近似函数常取易进行积分解析运算的低阶多项式。

数值计算 数值积分

数值积分的实现方法，主要关注：

一维空间中的积分（线积分）

规则的二维和三维空间中的积分（面积分和体积分）

梯形积分: 分段线性多项式（直线）代替被积函数

辛普森积分： 分段二次多项式（抛物线）代替被积函数

…

数值计算 数值积分

trapz 用于离散的被积函数点 (分段一次多项式代替被积函数)

向量x,y定义了函数关系y=f(x)

【例】 用trapz函数计算定积分

```
X=1:0.01:2.5;
Y=exp(-X);        %生成函数关系离散数据向量
trapz(X,Y)
          ans =
                0.28579682416393
```

lookfor trapz 查看trapz及其变种的详细用法。

数值计算 数值积分

quad
自适应步长辛普森积分 (分段二次多项式代替被积函数)

[I,n]=quad('fname',a,b,tol,trace)

fname是被积函数,由于M函数文件定义。
a和b是积分下限和上限
tol为积分精度，缺省时为1E-6
trace=0为不显示积分过程，非零为显示
I返回积分值，n返回被积函数调用次数

数值计算 数值积分

【例】 求f=exp(-0.5*x)*sin(x+pi/6)从0到3pi的积分

建立被积函数文件fesin.m

```
function f=fesin(x)
f=exp(-0.5*x).*sin(x+pi/6);
```

调用quad求定积分

```
[S,n]=quad('fesin',0,3*pi,1e-6)
S = 0.9008
n = 77 %如提高精度，则调用函数次数将增加
```

数值计算 数值积分

其他常用积分方法

- 牛顿-柯兹积分 quad8（新版已淘汰）
- 自适应高斯-拉巴托积分 quadl
- 自适应高斯-克罗诺德积分quadgk
- integral triplequad quadv …

以上是matlab自身集成的常用积分函数，还有更多积分算法需要编程实现。

数值计算 数值积分 二维

二重定积分的数值计算 dblquad函数

dblquad(f,a,b,c,d,tol,trace)

计算f(x,y)在[a,b]×[c,d]矩形区域上的二重定积分。参数tol，trace的用法与quad类似。

还有 quad2d，integral2 等二重积分，
以及 integral3，triplequad 等三重积分函数。

数值计算 数值积分 二维

【例】 计算二重定积分

(1) 建立函数文件fxy.m：

```
function f=fxy(x,y)
global ki;
ki=ki+1;          %ki用于统计被积函数的调用次数
f=exp(-x.^2/2).*sin(x.^2+y);
```

(2) 调用dblquad函数求解。

```
global ki; ki=0;
I=dblquad('fxy',-2,2,-1,1)
ki
I =   1.57449318974494
ki =   1038
```

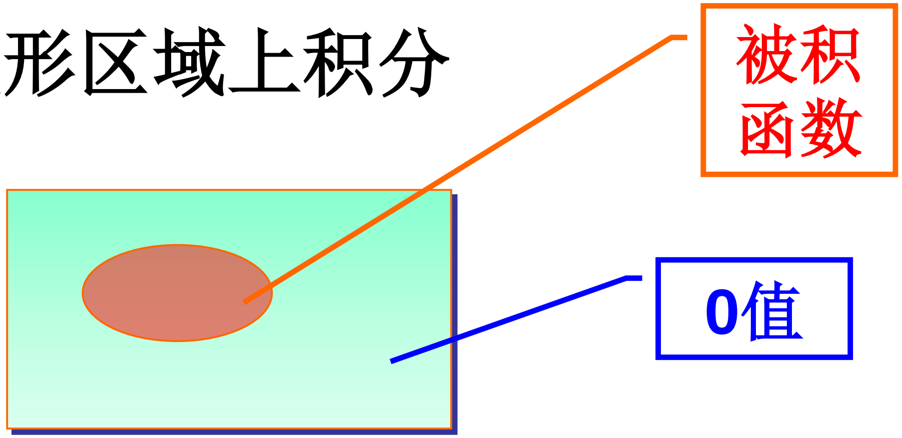
数值计算 数值积分 二维

dblquad 还可求解一般区域（非矩形）上的二重积分。

step1 把非矩形区域的被积函数外推到一个矩形区域，

step2 在原被积区域上，外推函数取原有值，而在外推部分取0

step3 对新函数在矩形区域上积分



从R2009a版本开始引入了专门求解一般区域二重积分的函数——**quad2d**，效率大大提高。

数值计算 数值积分 二维

【例】 求被积函数 $f(x,y)=\sqrt{10000-x^2}$ 在 区域 $x^2+y^2\le 10000$ 内的积分。

```
y1=dblquad(@(x,y) sqrt(10^4-x.^2).*(x.^2+y.^2<=10^4),
                    -100,100,-100,100)
        y1 =
                    2.6667e+006


y2=quad2d(@(x,y) sqrt(10^4-x.^2),-100,100,
                   @(x) -sqrt(10^4-x.^2),@(x) sqrt(10^4-x.^2))

        y2 =
                    2.6667e+006
```

补充内容

涉及微分，微分方程组，傅里叶分析、稀疏矩阵等，有兴趣的同学可以自学。

微分

利用差分函数计算数值微分

在matlab中，导数或微分的数值计算并没有对应的函数，但可使用向前差分函数diff进行间接计算。

DX=diff(X,n):

计算向量X的n阶向前差分，DX(i)=X(i+1)-X(i)，i=1,2,…,n-1

DX=diff(A,n,dim):

计算矩阵A的n阶差分，dim=1时(缺省)，按列计算差分；dim=2则按行计算差分。

此外，还可利用拟合或插值函数来计算导数或微分的近似值

微分

例 用多种方法计算函数**f(x)**的数值导数，并绘图比较

```
f=inline('sqrt(x.^3+2*x.^2-x+12)+(x+5).^(1/6)+5*x+2');
g=inline('(3*x.^2+4*x-1)./sqrt(x.^3+2*x.^2-
   x+12)/2+1/6./(x+5).^(5/6)+5');
x=-3:0.01:3;
p=polyfit(x,f(x),5);      %用5次多项式p拟合f(x)
dp=polyder(p);            %对拟合多项式p求导数dp
dpx=polyval(dp,x);        %求dp在假设点的函数值
dx=diff(f([x,3.01]))/0.01;   %直接对f(x)求数值导数
gx=g(x);      %求函数f的导函数g在假设点的导数
plot(x,dpx,x,dx,'.',x,gx,'-');   %作图
```

傅立叶分析

- Y=fft(X,n,dim) 离散傅立叶变换

(1)当X是一个向量时，返回对X的离散傅立叶变换。

(2)当X是一个矩阵时，返回一个矩阵并送Y，其列(行)是对X的列(行)的离散傅立叶变换。

- Y=ifft(X,n,dim)

离散傅立叶变换的逆变换，其中X、n、dim的意义及用法和fft相同。

傅立叶分析

例 对矩阵A的列向量、行向量分别进行离散傅立叶变换、并对变换结果进行逆变换。

命令如下：

```
A=[3,2,1,1;-5,1,0,1;3,2,1,5];
fftA=fft(A)            %求A的列向量的傅立叶变换
fftA2=fft(A,4,2)       %求A的行向量的傅立叶变换
ifft(fftA)             %对矩阵fftA的列向量进行傅立叶逆
                         变换，结果应等于A
ifft(fftA2,4,2)        %对矩阵fftA2的行向量进行傅立
                         叶逆变换，其结果应等于A
```

傅立叶分析

```
fftA =
   1.0000              5.0000              2.0000              7.0000
   4.0000 + 6.9282i   0.5000 + 0.8660i   0.5000 + 0.8660i  -2.0000 + 3.4641i
   4.0000 - 6.9282i   0.5000 - 0.8660i   0.5000 - 0.8660i  -2.0000 - 3.4641i

fftA2 =
   7.0000              2.0000 - 1.0000i   1.0000              2.0000 + 1.0000i
  -3.0000             -5.0000             -7.0000             -5.0000
  11.0000              2.0000 + 3.0000i  -3.0000              2.0000 - 3.0000i

ans =
    3.0000    2.0000    1.0000    1.0000
   -5.0000    1.0000    0.0000    1.0000
    3.0000    2.0000    1.0000    5.0000
ans =
     3     2     1     1
    -5     1     0     1
     3     2     1     5
```

稀疏矩阵

矩阵的完全存储模式

完全存储方式是将矩阵的全部元素按列存储。以前讲到的矩阵的存储方式都是按这个方式存储的，此稀疏存储方式存储方式对稀疏矩阵也适用.

稀疏矩阵的存储方式

仅存储矩阵所有的非零元素的值及其位置，即行号和列号。在MATLAB中，稀疏存储方式也是按列存储的。

- 在讲稀疏矩阵时，有两个不同的概念，一是指矩阵的0元素较多，该矩阵是一个具有稀疏特征的矩阵，二是指采用稀疏方式存储的矩阵。

稀疏矩阵

稀疏存储方式的产生与转化

1. 将一个完全存储方式的转化为稀疏存储方式

B=sparse(A)

将矩阵A转化为稀疏存储方式的矩阵B。

sparse函数还有其他一些格式：

sparse(m,n) 生成一个m×n的所有元素都是0的稀疏矩阵。

sparse(u,v,S) u、v、S是三个等长的向量。

此外，还有一些和稀疏矩阵操作有关的函数。例如

[U,V,S]=find(A) 返回矩阵A中非0元素的下标和元素。这里产生的U、V、S可作为sparse(u,v,s)的参数。

full(A) 返回和稀疏存储矩阵A对应的完全存储方式矩阵。

稀疏矩阵

2. 产生一个稀疏矩阵

把要建立的稀疏矩阵的非0元素及其所在行和列的位置表示出来后由MATLAB自己产生其稀疏存储方式，这需要使用spconvert函数。调用格式为：

B=spconvert(A)

其中A为一个m×3或m×4的矩阵，其每行表示一个非0元素，m是非0元素的个数。

3. 单位稀疏矩阵的产生

单位矩阵只有对角线元素为1，其他元素都为0，是一种具有稀疏特征的矩阵。我们知道，函数eye产生一个完全存储方式的单位矩阵。MATLAB还有一个产生稀疏存储方式的单位矩阵的函数，这就是speye。函数speye(m,n)返回一个m×n的稀疏存储单位矩阵。

稀疏矩阵

稀疏存储矩阵只是矩阵的存储方式不同，它的运算规则与普通矩阵是一样的。所以，在运算过程中，稀疏存储矩阵可以直接参与运算。当参与运算的对象不全是稀疏存储矩阵时，所得结果一般是完全存储形式。

常微分方程组

1、常微分方程的一般形式：

n 阶

$$F(x, y, y',\dots,y^{(n)}) = 0$$ *隐式*

或 $$y^{(n)} = f(x, y, y',\dots,y^{(n-1)})$$ *显式*

特例：

1 阶

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y, y')$$ 初始条件：$y(t_0) = y_0$

2、一阶常微分方程组的一般形式：

$$\begin{cases} \dfrac{dy_1}{dx} = f_1(x, y_1(x), y_2(x), \cdots, y_m(x)); \\ \cdots\cdots \\ \dfrac{dy_m}{dx} = f_m(x, y_1(x), y_2(x), \cdots, y_m(x)); \end{cases}$$

常微分方程组

常微分方程解的形式

简单系统

解析解 $y = f(x)$

复杂/大型系统

数值解 $(x_i\ \ y_i)$

常微分方程组

用matlab求常微分方程解析解

一阶常微分方程: $\dfrac{\mathrm{d}y}{dx} = f(x, y)$

获取解析解的方法:

① 分离变量法

② 齐次方程的变换法;

……

常微分方程组

dsolve('eqn1','eqn2', ..., ' c1 ',..., ' var1 ',...)

微分方程组

初值条件

变量组

y' ⟺ Dy， y'' ⟺ D2y

自变量名可以省略，默认变量名't'

常微分方程组

例1： $$\frac{dy}{dx}=1+y^2, \qquad y(0)=1$$

输入： y=dsolve ('Dy=1+y^2')

y1=dsolve('Dy=1+y^2','y(0)=1','x')

输出： y= tan(t-C1) （通解，一簇曲线）

y1= tan(x+1/4*pi)（特解，一条曲线）

常微分方程组

例2：常系数的二阶常微分方程

$$y''-2y'-3y=0, \qquad y(0)=1, y'(0)=0$$

```
y=dsolve('D2y-2*Dy-3*y=0','x')
```

```
y=dsolve('D2y-2*Dy-3*y=0','y(0)=1,Dy(0)=0','x')
```

例3：变系数的二阶常微分方程

$$x''(t)-(1-x^2(t))x'(t)+x(t)=0, \qquad x(0)=3, x'(0)=0$$

```
y=dsolve('D2x-(1-x^2)*Dx+x=0', ' x(0)=3,Dx(0)=0')
```

有些方程并无解析解，如例3

常微分方程组

例4:　非线性常微分方程

$$x'(t)^2 + x(t)^2 = 1, x(0) = 0$$

```
x=dsolve('(Dx)^2+x^2=1', 'x(0)=0')
```

```
x =   [  sin(t) ]
      [ -sin(t) ]
```

某个特定数值解，如何得到?

```
t=pi/2;
eval(x)
```

常微分方程组

例5: $$\begin{cases} \dfrac{dx}{dt} = 3x + 4y \\ \dfrac{dy}{dt} = -4x + 3y \end{cases} \qquad \begin{cases} x(0) = 0 \\ y(0) = 1 \end{cases}$$

[x,y]=dsolve('Dx=3*x+4*y','Dy=-4*x+3*y')

[x,y]=dsolve('Dx=3*x+4*y','Dy=-4*x+3*y', 'x(0)=0,y(0)=1')

结果

x =   exp(3*t)*(C1*sin(4*t)+C2*cos(4*t))

y = - exp(3*t)*(-C1*cos(4*t)+C2*sin(4*t))

MATLAB数值计算 常微分方程组

数值解
非常重要

1、欧拉法

2、龙格—库塔法…

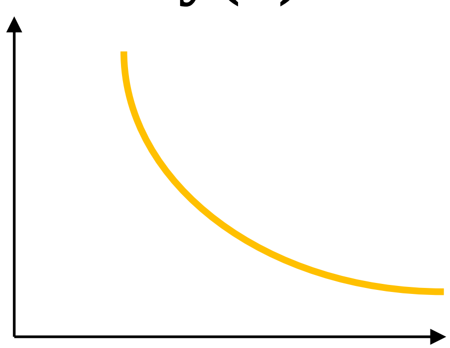
$$\frac{dy}{dx}=f(x,y),\qquad y(x_0)=y_0$$

数值求解思想：变量离散化

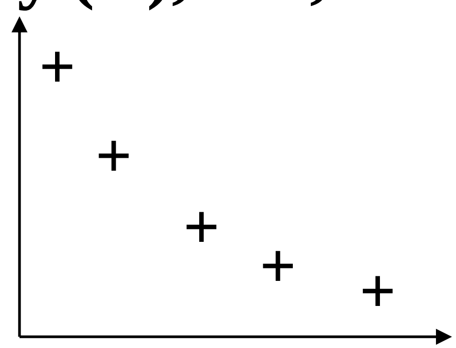
连续

离散

y(x)



$y_i(x_i)$, i=1,2...n



常微分方程组

$[t, x] = ode23('f', ts, x_0, options)$

自变量值

函数值

或
ode45
ode113
ode15s
ode23s

待解方程
函数文件

$ts=[t_0, t_f]$，t_0、t_f为自变量的初值和终值

函数的初值

ode23：组合的2、3阶龙格-库塔算法
ode45：组合的4、5阶龙格-库塔算法

设定误差容限(缺省时相对误差10^{-3}, 绝对误差10^{-6}),
需要提前设定：options=odeset（'reltol',rt,'abstol',at），
rt，at：为相对误差和绝对误差.

常微分方程组

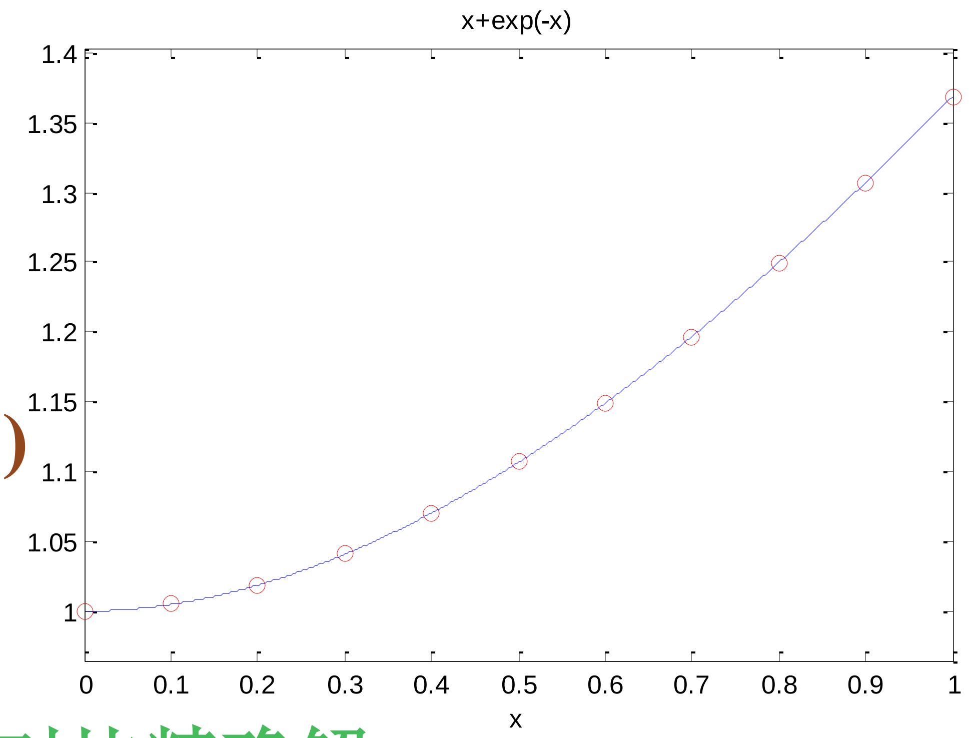
例：　$y' = -y + x + 1$，$y(0) = 1$

该问题有精确解析解x+exp(-x)

M-文件 weif.m

```
function f = weif(x,y)
f=-y+x+1;
```

```
[x, y]=ode23('weif', [0, 1], 1)
plot(x, y, 'r');
hold on
ezplot('x+exp(-x)',[0, 1]) %对比精确解
```



常微分方程组

常微分方程组的数值求解

1、解n个未知函数的方程组，x_0和x均为n维向量，m-函数文件中的待解方程组应以x的分量形式写成.

2、使用Matlab求数值解时，高阶微分方程须等价变换成一阶

常微分方程组

常微分方程组 $$\begin{aligned}\frac{dx(t)}{dt} &= f_1(t, x(t), y(t)) \\ \frac{dy(t)}{dt} &= f_2(t, x(t), y(t))\end{aligned}$$

step1、建立函数文件

```
function  xdot = fun(t,x,y)
    xdot = [f1(t, x(t), y(t))； f2(t, x(t), y(t))]；
```

step2、数值计算（执行以下命令）

```
[t, x, y]=ode23('fun', [t0, tf], [x0, y0])
```

常微分方程组

高阶常微分方程 $y'' = f(y', y, t)$

令 $y' = x$ ，因此原方程记为 $\begin{cases} x' = f(x, y, t) \\ y' = x \end{cases}$

y(t)是原方程解
x(t)是中间变量

函数文件

function xdot = fun1(t,*x*,*y*)　（fun1.m)

xdot = [*f*(t, *x*(t), *y*(t))；*x*(t)]；

$[t, x, y] =$ ode23(‘fun1’, $[t_0, t_f]$, $[x_0, y_0]$)

类似地，含有更高阶导数的方程也能求解

常微分方程组

是否有解析解？
尝试 dsolve

例　**求解Van der pol 方程：**

$$x''(t)-(1-x(t)^2)x'(t)+x(t)=0$$

$$x(0)=3, x'(0)=0$$

令 $y_1=x(t)$，$y_2=x'(t)$;

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

$$y_1(0)=3, \qquad y_2(0)=0;$$

常微分方程组

Step1:编写M文件 (文件名为 vdpol.m):

```
function yp = vdpol(t,y);
yp=[y(2);(1-y(1)^2)*y(2)-y(1)];
```

Step2:编写程序如下:（vdj.m）

```
[t,y]=ode23('vdpol',[0,20],[3,0]);
y1=y(:,1);   % 原方程的解
y2=y(:,2);
 plot(t,y1,t,y2,‘--’)    % y1(t),y2(t) 曲线图
 pause,
 plot(y1,y2),grid, % 相轨迹图，即y2(y1)曲线
```

常微分方程组

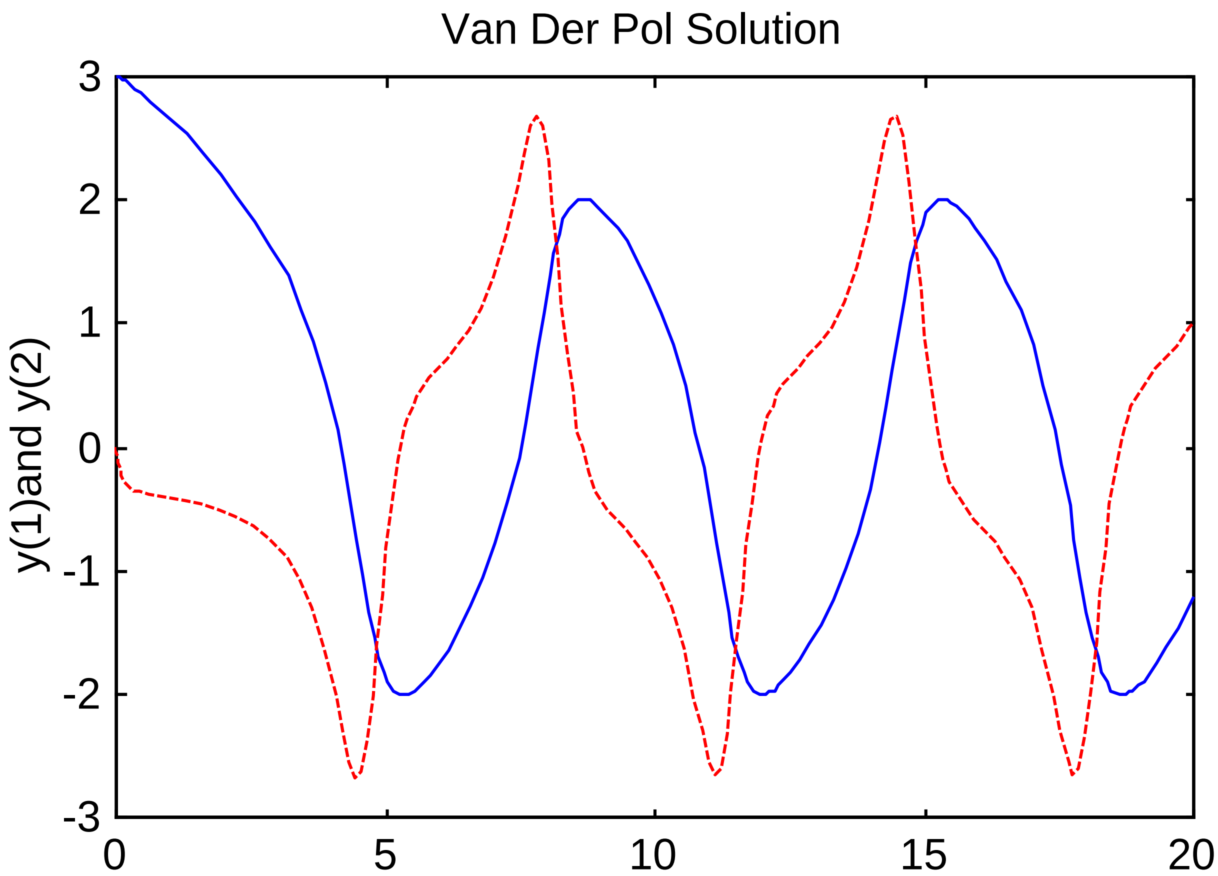
计算结果

蓝线 y_1

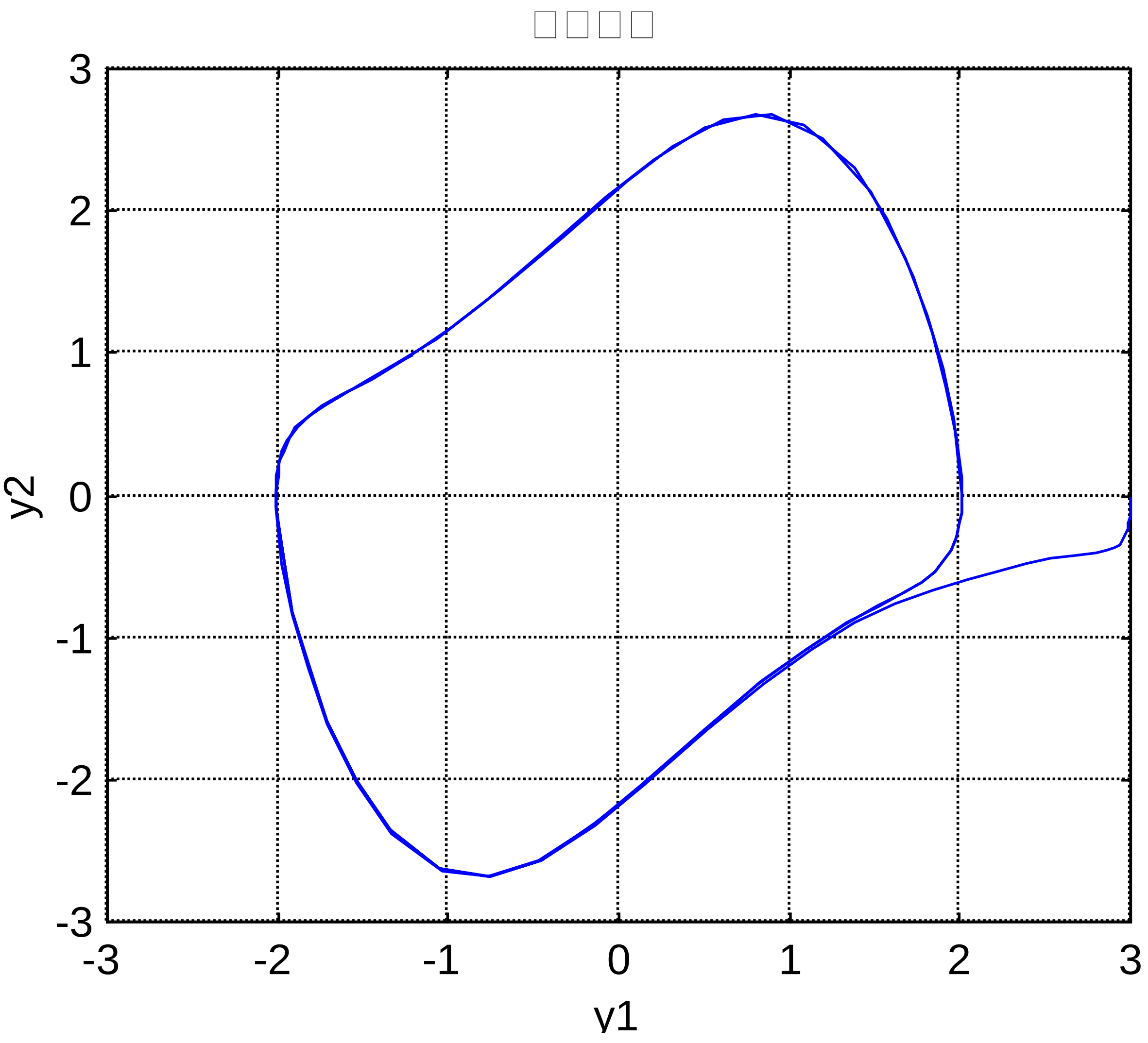
（原方程解

红线 y_2

（y_1的导数



常微分方程组



常微分方程组

例　　　求解Lorenz模型：

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

其中β=8/3，σ=10，ρ=28

Step1: 编写M函数文件(lorenz.m)

Step2: 数值求解并画三维空间的相平面轨线(ltest.m)

常微分方程组

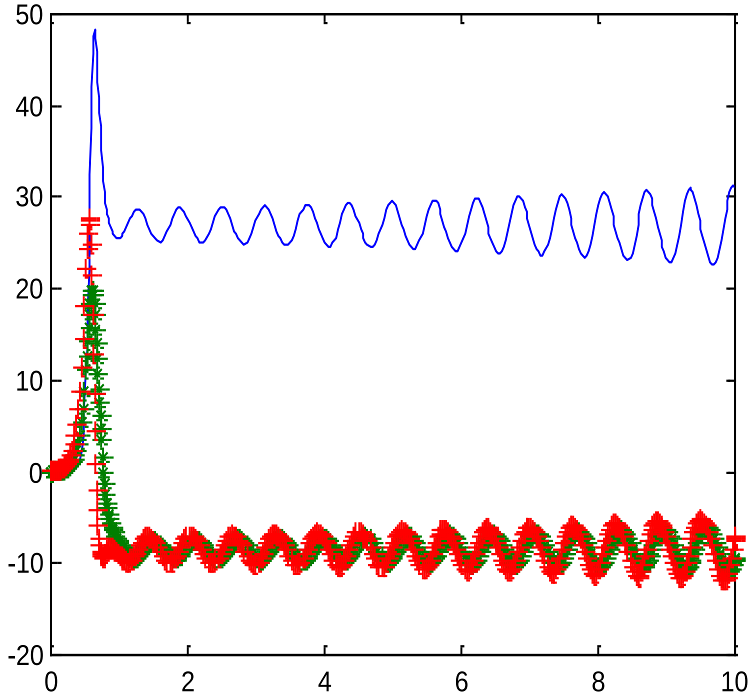
函数文件lorenz.m

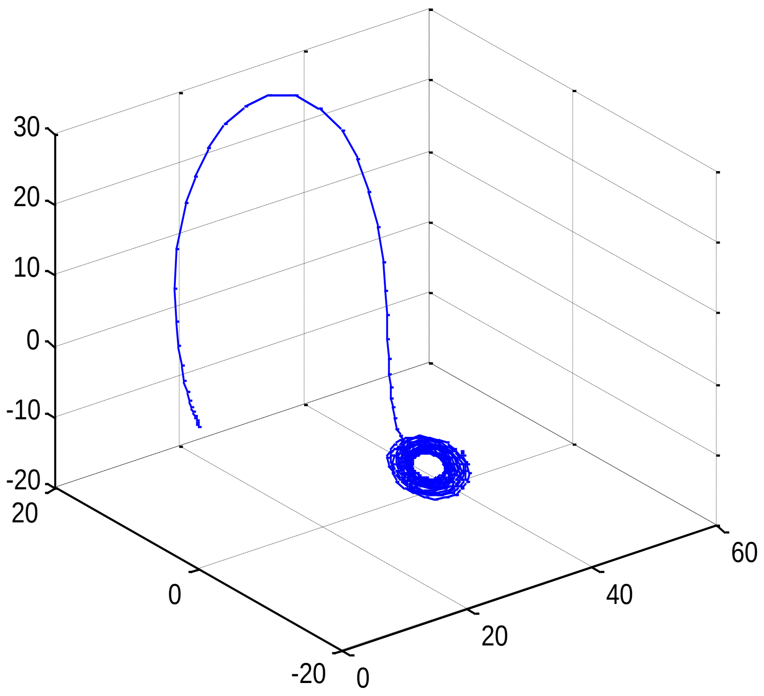
```
function xdot=lorenz(t,x)
xdot=[-8/3,0,x(2);0,-10,10;-x(2),28,-1]*x;
```

命令文件ltest.m

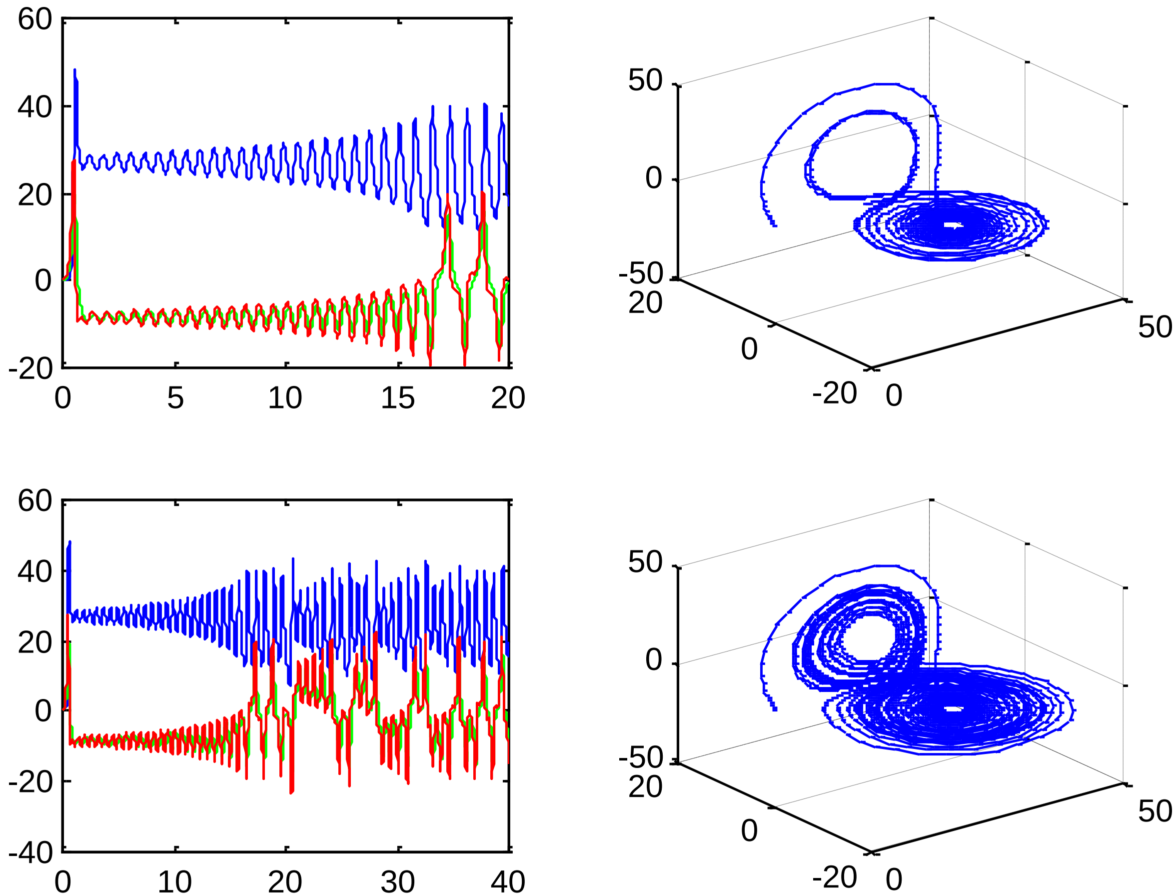
```
x0=[0 0 0.1]';
[t,x]=ode45('lorenz',[0,10],x0);
plot(t,x(:,1),'-',t,x(:,2),'*',t,x(:,3),'+')
pause
plot3(x(:,1),x(:,2),x(:,3)),grid on
```

常微分方程组





常微分方程组



曲线趋于振荡发散

三维图形趋于混沌

常微分方程组

偏微分方程（组）如何解？

例如：$$\frac{dw}{dt}+\frac{df(w)}{dx}=0$$

极少数特殊、简单形式的偏微分方程有解析解；

大部分**PDE**需要数值求解；

常用思路是将**PDE**近似地变换为**ODE**；

第5章，数值计算

1. 多项式 roots，poly，polyval

2. 解线性方程组 A\b的运用

3. 解非线性方程组 solve fsolve

4. 插值与拟合 interp1 interp2 polyfit

5. 数值积分 trapz

掌握以上数值计算函数的基本用法，了解其主要设置参数。

适当了解其他内容。**596802484**