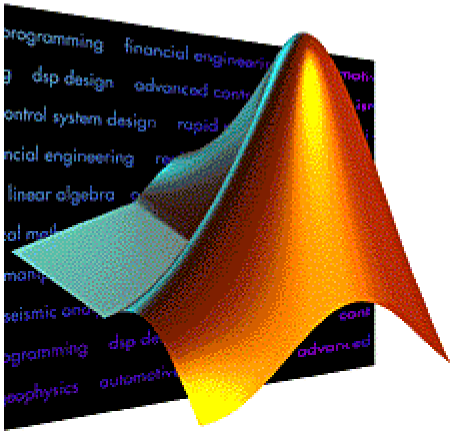
Matlab 程序设计与应用
第6章：Matlab数值计算

伍振海



September 24, 2017

Autumn, @swpu

第6章 数据统计处理

本章要点

6.1 数据处理与多项式计算

6.2 数值微积分

6.3 离散傅立叶变换

6.4 线性方程组求解

6.5 常微分方程数值求解

6.1.1 数据统计处理：最大值与最小值

- 最大值：max，最小值：min，用法相同！
- 向量最大值和最小值：
  - y=max(X)：向量X的最大值
  - [y,I]=max(X)：向量X的最大值(y)及其序号(I)

    ——复数元素按模取最大值

例6-1 求向量x=[-43,72,9,16,23,47]的最大值。

```
x=[-43,72,9,16,23,47];
y=max(x)              %求向量x中的最大值
[y,l]=max(x)          %求最大值及其位置
```

6.1.1 数据统计处理：最大值与最小值

- 求矩阵的最大值和最小值
  - **Y=max(A)**：取每一列的最大值,返回给行向量Y
  - **[Y,U]=max(A)**：取每一列的最大值及其行号
  - **[Y,U]=max(A,[],dim)**：
    dim=1时，取每列最大值，相当于max(A)
    dim=2时，取每行最大值
  - 取整个矩阵的最大值：
    max(max(A))
    max(A(:))

```
A=rand(5);
Y=max(A)
[Y,U]=max(A,[],1)
Y=max(A,[],2)
Y=max(A(:))
```

6.1.1 数据统计处理：最大值与最小值

- 两个向量或矩阵对应元素的比较
  - ➢**U=max(A,B)**：对应元素作比较，取较大者
  - ➢**U=max(A,n)**：每个元素与n比较，取较大者

例: 求两个2×3矩阵x, y所有同一位置上的较大元素构成的新矩阵p。

```
x=[4,5,6;1,4,8];
y=[1,7,5;4,5,7];
p=max(x,y)
q=max(x,4.5)
```

- 思考：为什么取矩阵每一行的最大值语法为：**max(A,[],dim)**，而不直接用**max(A,dim)**？

6.1.1 数据统计处理：平均值与中值

- 平均值与中值的概念
- 平均值：mean

m=mean(X)： 返回向量X的算术平均值m。

Y=mean(A)：求矩阵A每一列的平均值，
返回给行向量Y。

Y=mean(A,dim)：

dim=1时，取每列平均值，相当于mean(A)
dim=2时，取每行平均值

- 中值：median

用法与mean相同！

6.1.1 数据统计处理：求和与求积

- 求和:sum

s=sum(X)：返回向量**X**各元素的和**s**。

S=sum(A)：求矩阵**A**每一列元素之和，并返回给行向量**S**。

S=sum(A,dim)：

dim=1：求每一列元素之和，等同于**sum(A)**；

dim=2：求每一行元素之和，并返回一个列向量。

- 求积：**prod**

用法与**sum**相同

6.1.1 数据统计处理：累加与累积

- 累加：**cumsum**
- 累乘：**cumprod**
  - ➢用法和sum等函数类似！

x	cumsum(x)	cumprod(x)
$x1$	$x1$	$x1$
$x2$	$x1+x2$	$x1\times x2$
$x3$	$x1+x2+x3$	$x1\times x2\times x3$
$\cdots$	$\cdots$	$\cdots$

6.1.1 数据统计处理：标准方差与相关系数

- 标准方差：**Y=std(A,flag,dim)**

  dim=1：求各列元素的标准方差；（缺省值）

  dim=2：求各行元素的标准方差。

  flag：计算公式，取0或1：（p143)

  flag=0：按S1所列公式计算 （缺省值）

  flag=1：按S2所列公式计算。

- 相关系数：**corrcoef**

  corrcoef(A)：返回从矩阵X形成的一个相关系数矩阵。此相关系数矩阵的大小与矩阵X一样。它把矩阵X的每列作为一个变量，然后求它们的相关系数。

6.1.1 数据统计处理：标准方差与相关系数

例6-5　生成满足正态分布的10000×5随机矩阵，然后求各列元素的均值和标准方差，再求这5列随机数据的相关系数矩阵。

```
X=randn(10000,5);
M=mean(X)
D=std(X)
R=corrcoef(X)
```

6.1.1 数据统计处理：元素排序

- 排序：**sort**
- **sort(X)**：返回一个升序排列的新向量。
- **[Y,I]=sort(A,dim,mode)**
  - **Y**：排序后的矩阵
  - **I**：记录**Y**中的元素在**A**中位置

参数：

- **dim=1**：按列排序
- **dim=2**：按行排序
- **mode**：升序或降序

  'ascend'：升序

  'descend'：降序

6.1.4 多项式: 表示

- 许多函数都可以展开为高阶多项式

$$f(x)=P_n x^n+P_{n-1}x^{n-1}+....P_1 x+P_0$$

- **Matlab** 使用一个向量来表示多项多的系数
  - 如：使用向量**P**来表示一个多项式：

ax³+bx²+cx+d

$$\mathbf{ax^3+bx^2+cx+d}$$

P(1)　P(2)　P(3)　P(4)

  - **P=[1 0 -2]**　表示多项式：$\mathbf{x^2-2}$
  - **P=[2 0 0 0]** 表示多项式：$\mathbf{2x^3}$

6.1.4 多项式运算：四则运算

- **多项式的加减运算：** 按对应系数向量的加减运算
- **多项式乘法：** **conv(P1,P2)**

例：求多项式x^4+8x^3-10与$2x^2-x+3$之和、积。

```
P1=[1,8,0,0,-10];
P2=[0,0,2,-1,3];
PS=P1+P2;
PC=conv(P1,P2);
```

```
PS =
     1     8     2    -1    -7
PC =
     0     0     2    15    -5    24   -20    10   -30
```

6.1.4 多项式运算：四则运算

- 多项式除法：
  - [Q,r]=deconv(P1,P2)
    - Q: 多项式P1除以P2的商式
    - r: 多项式P1除以P2的余式。
    - Q 和r仍是多项式系数向量。

- deconv是conv的逆函数，即有

$$P1=conv(P2,Q)+r$$

6.1.4 多项式运算：多项式求值

$$f(x) = P_n x^n + P_{n-1} x^{n-1} + \ldots P_1 x + P_0$$

- 计算多项式在某一点的值：

```
» y0=polyval(P,x0)
```

  - x0 与 y0 都是单个值

- 计算多项式在多个点的值：

```
» y=polyval(P,x)
```

  - 对向量或矩阵中的每个元素求其多项式的值
  - x 与 y 是相同大小的向量或矩阵！
  - 求多项式的导数：

```
» K=polyder(p)
```

6.1.4 多项式运算：多项式求根

- **P** 是一个长度为**N+1**的向量，表示一个**N**阶多项式
- 获得多项式的根

```
» r=roots(P)
```

  - **r** 是一个长度为**N**的向量

- 如果已知多项式的全部根，可以由根来构建对应的多项式

```
» P=poly(r)
```

  - **r** 是一个长度为**N**的向量

6.1.4 多项式运算：多项式求根

例6-22 已知 $f(x)=3x^5+4x^3-5x^2-6.1x+5$

(1) 计算f(x)=0 的全部根。(2) 由方程f(x)=0的根构造一个多项式g(x)，并与f(x)进行对比。

```
P=[3,0,4,-5,-6.1,5];
X=roots(P)              %求方程f(x)=0的根
G=poly(X)               %求多项式g(x)
```

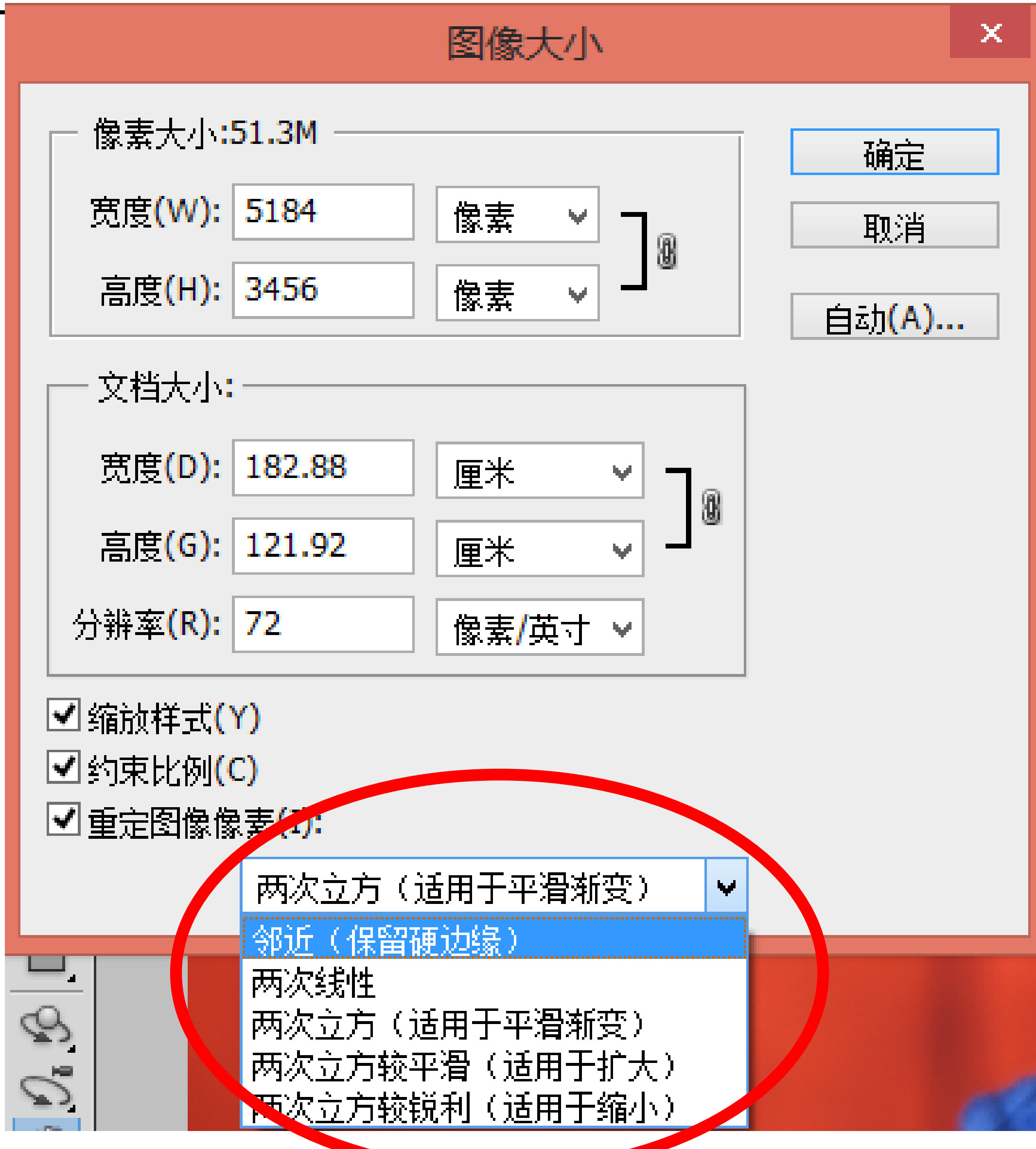
```
X =
  -0.3207 + 1.5991i
  -0.3207 - 1.5991i
  -0.9695 + 0.0000i
   0.8549 + 0.0000i
   0.7559 + 0.0000i
```

```
G =
1.0000   -0.0000    1.3333   -1.6667   -2.0333    1.6667
```

6.1.2 数据插值

- 摄像头的像素问题
- 图像处理

●插值:在离散数据的基础上补插连续函数，使得这条连续曲线通过全部给定的离散数据点。插值是离散函数逼近的重要方法，利用它可通过函数在有限个点处的取值状况，估算出函数在其他点处的近似值。



6.1.2 数据插值：一维数据插值

- 一维数据插值：**interp1**

Y1=interp1(X,Y,X1,'method')

- **X,Y**: 两个等长已知向量，分别描述采样点和样本值
- **X1**: 一个向量或标量，描述欲插值的点
- **Y1**:是一个与**X1**等长的插值结果。

➢ 注意：**X1**的取值范围不能超出**X**的给定范围，否则，会给出“**NaN**”错误。

6.1.2 数据插值：一维数据插值

- **method：** 插值方法：
  - **‘linear’：** 线性插值→缺省值
    两点间作直线，取对应点数据
  - **‘nearest’：最近点插值**
    取最近点的数据点进行插值
  - **‘cubic’：3次多项式插值**
    生成3次多项式，由该式决定插值点数据
  - **‘spline’：3次样条插值**
    每个分段内构造一个3次多项式进行插值
  - **3次样条插值函数：Y1=spline(X,Y,X1)**
    等效于：**Y1=interp1(X,Y,X1,‘spline’)**

6.1.2 数据插值：一维数据插值

	运算时间	占用内存	光滑程度
最近点插值：	快	少	差
线性插值：	稍长	较多	稍好
三次多项式插值：	较长	多	较好
三次样条插值：	最长	较多	最好

例6-7 用不同的插值方法计算f(x)在0.472点的值 (p146)

```
x=0.46:0.01:0.49;
f=[0.484655,0.4937542,0.5027498,0.5116683];
interp1(x,f,0.472)
interp1(x,f,0.472,'spline')
...
```

6.1.2 数据插值：一维数据插值

例：某观测站测得某日6:00时至18:00时之间每隔2小时的室内外温度(℃)，用3次样条插值分别求得该日室内外6:30至17:30时之间每隔2小时各点的近似温度(℃)。

h(时间):	**6**	**8**	**10**	**12**	**14**	**16**	**18**
t1(室内):	**18**	**20**	**22**	**25**	**30**	**28**	**24**
t2(室外):	**15,**	**19**	**24**	**28**	**34**	**32**	**30**

```
h =6:2:18;
t=[18,20,22,25,30,28,24;
    15,19,24,28,34,32,30]';
XI =6.5:2:17.5
YI=interp1(h,t,XI,‘spline’)    %3次样条插值
```

6.1.2 数据插值：二维数据插值

- 二维插值：interp2

  Z1=interp2(X,Y,Z,X1,Y1,'method')

  - 用法与参数与inerp1相同
  - 采样点由一维(X)变为二维(X,Y)，即：
    - X,Y：分别描述两个参数的采样点
    - Z：与参数采样点对应的函数值
    - X1,Y1：两个向量或标量，描述欲插值的点
    - Z1：根据相应的插值方法得到的插值结果
  - X,Y,Z可以是向量形式，也可以是矩阵形式
    若为向量形式，会自动使用meshgrid生成矩阵形式来进行计算

6.1.2 数据插值: 二维数据插值

例6-9 设$z=x^2+y^2$，对z函数在$[0,1]\times[0,2]$区域内进行插值。**(p148)**

```
x=0:0.1:1;
y=0:0.2:2;
[X,Y]=meshgrid(x,y);
Z=X.^2+Y.^2;
interp2(X,Y,Z,[0.5 0.6],[0.4 0.5])

%  等价于:
interp2(x,y,Z,[0.5 0.6],[0.4 0.5])
```

6.1.2 数据插值：二维数据插值

例6-10　某实验对一根长10米的钢轨进行热源的温度传播测试。用x表示测量点0:2.5:10(米)，用h表示测量时间0:30:60(秒)，用T表示测试所得各点的温度(℃)。试用线性插值求出在一分钟内每隔20秒、钢轨每隔1米处的温度TI。(p149)

```
x=0:2.5:10;
h=[0:30:60]';
T=[95,14,0,  0, 0;
    88,48,32,12,6;
    67,64,54,48,41];
xi=[0:10];
hi=[0:20:60]';
TI=interp2(x,h,T,xi,hi)
```

6.1.3 曲线拟合

- **曲线的多项式拟合：** 根据已知采样点的数据将其拟合成一个高阶的多项式表达式

- **polyfit：** 求最小二乘拟合多项式的系数

  [P,S]=polyfit(X,Y,m)

  - **X:** 采样点
  - **Y:** 采样点函数值
  - **m:** 多项式次数
  - **P:** 多项式系数，是一个长度为**m+1**的向量
  - **S:** 误差向量**S**

- **Y=polyval(P,X)**

  按所得多项式**P**计算所给点**X**上的函数近似值。

6.1.3 曲线拟合

例6-11(p150) 在[0,2pi]区间使用三次多项式拟合sin(*x*)

```
%拟合
X=linspace(0,2*pi,50);
Y=sin(X);
P=polyfit(X,Y,3);  %换成5如何？

%作图
X=linspace(0,2*pi,20);
Y=sin(X);
Y1=polyval(P,X);
plot(X,Y,'r-o',X,Y1,'b-*');
```

6.2.1 数值微分与求导

- **MATLAB中，没有直接求数值导数的函数，只有计算向前差分的函数diff**
- **DX=diff(X)：计算向量X的向前差分**
  - **DX(i)=X(i+1)-X(i)，i=1,2,…,n-1**
  - **Note:差分后的结果diff(x)，比X向量少一个元素！**

exa:

```
x=1:5
x =
     1     2     3     4     5
>> dx=diff(x)
dx =
     1     1     1     1
```

6. 2.1 数值微分与求导

- **DX=diff(X,n)**：计算X的n阶向前差分

  例如，**diff(X,2)=diff(diff(X))**。少几个元素？

- **DX=diff(A,n,dim)**：计算矩阵A的n阶差分

  dim=1：按列计算差分(缺省值)

  dim=2：按行计算差分

怎么求导数？

$$\frac{dy}{dx}=\frac{\Delta y}{\Delta x}=\frac{y(x+h)-y(x)}{(x+h)-x}=\frac{diff\,(\mathrm{y})}{h}=\frac{diff\,(\mathrm{y})}{diff\,(x)}$$

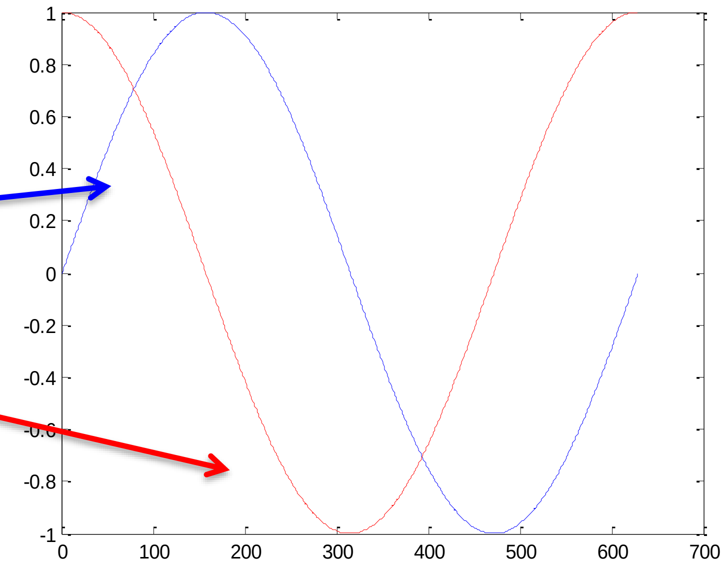
Note: 应尽量避免使用数值导数！

6.2.1 数值微分与求导

$$\frac{dy}{dx}=\frac{\Delta y}{\Delta x}=\frac{y(x+h)-y(x)}{(x+h)-x}=\frac{diff(y)}{h}=\frac{diff(y)}{diff(x)}$$

e.g.:求sin(x)及其导数

```
x=0:0.01:2*pi;
y=sin(x);
dydx=diff(y)./diff(x);
% plot(x,y,x,dydx);
plot(x,y,x(1:end-1),dydx)
```



6.2.1 数值微分与求导

例6.19 (p157) 用不同的方法求函数

$$f(x)=5x+(x+5)^{\frac{1}{6}}+\sqrt{x^3+2x^2-x+12}+2$$

在[-3,3]的数值导数，并做出f'(x)的图像。

法一：手动求出导数表达式，再代入求值

$$g(x)=f'(x)=\frac{1}{6(x+5)^{\frac{5}{6}}}+\frac{3x^2+4x-1}{2\sqrt{x^3+2x^2-x+12}}+5$$

法二：使用多项式求导：**polyder**

法三：使用数值求导方法：**diff(y)/diff(x)**

6.2.1 数值微分与求导

函数的定义： $f(x)=5x+(x+5)^{\frac{1}{6}}+\sqrt{x^3+2x^2-x+12}+2$

法一：使用函数文件，保存为f.m：

```
function y=f(x)
y=sqrt(x.^3+2*x.^2-x+12)+(x+5).^(1/6)+5*x+2;
```

法二：使用内联函数（不用另建函数文件）

```
f=inline('sqrt(x.^3+2*x.^2-x+12)+(x+5).^(1/6)+5*x+2');
```

法三：使用匿名函数（不用另建函数文件）

```
f=@(x) sqrt(x.^3+2*x.^2-x+12)+(x+5).^(1/6)+5*x+2;
```

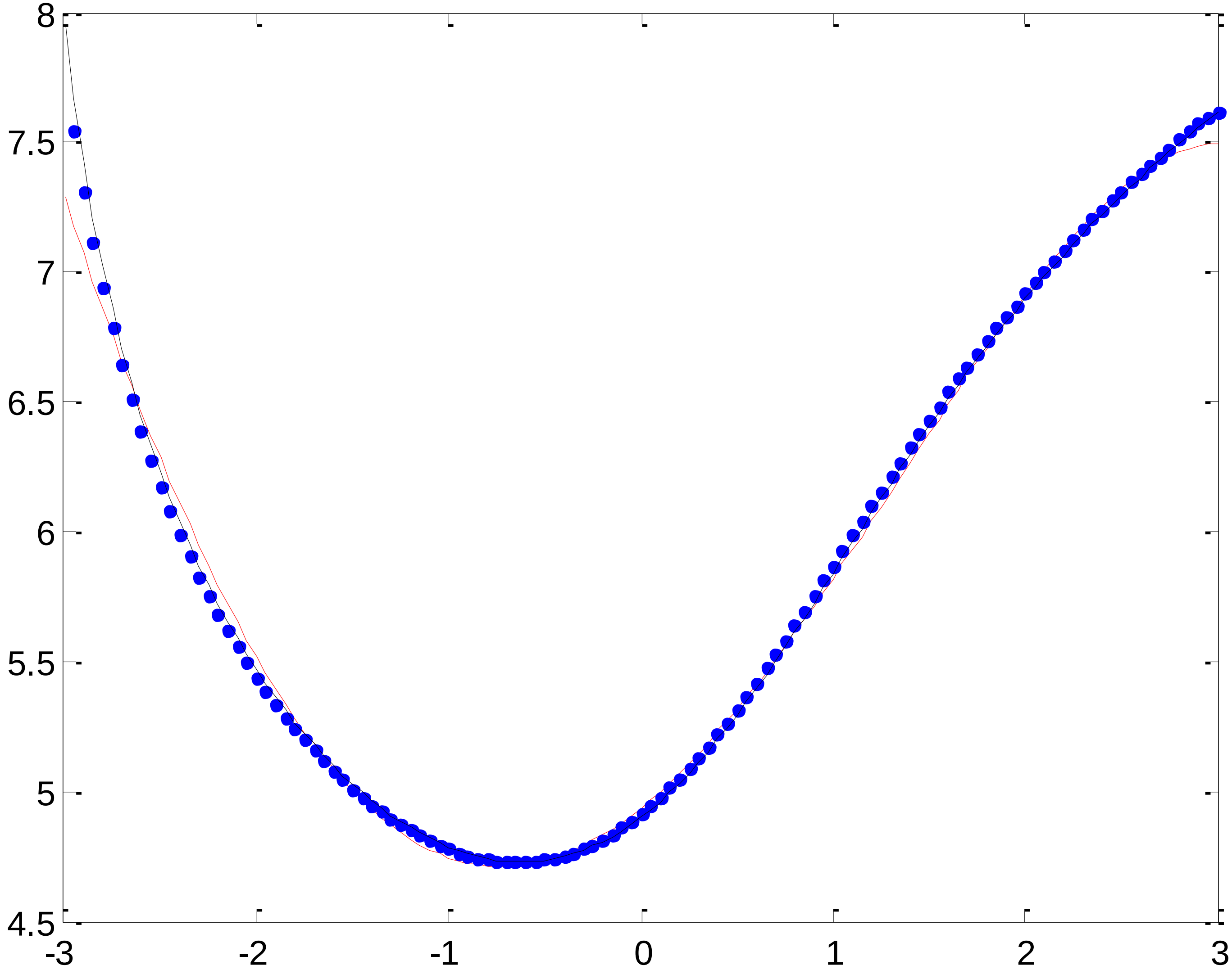
变量　　　　函数

查看二者区别：**whos f**

6.2.1 数值微分与求导

```
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'); %手动求导数
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;   %对f(x)求数值导数
   %等效于：dx=diff(f([x,3.01]))./diff([x,3.01]);
gx=g(x); %求函数f的导函数g在假设点的导数
plot(x,dpx,'r',x,dx,'b.',x,gx,'k');   %作图
```

6.2.1 数值微分与求导

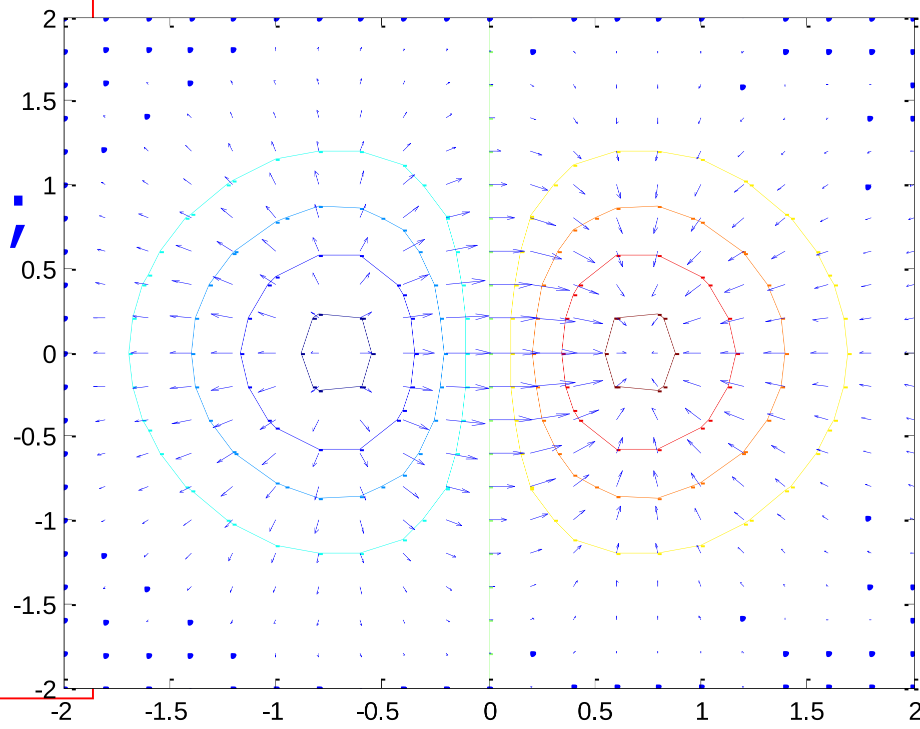


6.2.1 数值微分与求导

● 二维梯度：

➢[dx,dy]=gradient(mat);

```
v = -2:0.2:2;
[x,y] = meshgrid(v);
z = x .* exp(-x.^2 - y.^2);
[px,py] = gradient(z);
contour(v,v,z)
hold on
quiver(v,v,px,py)
hold off
```



6.2.2 数值积分

- 数值积分基本原理

　　将整个积分区间[a,b]分成n个子区间[x_i，x_{i+1}]，i=1,2,…,n，其中$x_1=a$，$x_{n+1}=b$。这样求定积分问题就分解为求和问题。

- 求解定积分的数值方法:

　　简单的梯形法、辛普生(Simpson)法、牛顿－柯特斯(Newton-Cotes)法等。

6.2.2 数值积分

- 变步长辛普生法

➢自适应simpson:

积分值

被积函数的调用次数

是否展现积分过程:
0： 不展现(缺省值)
非0: 展现

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

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

被积函数名

积分下限

上限

容差：积分精度

➢自适应Labatto:

[I,n]=quadl(@fname,a,b,tol,trace)

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

6.2.2 数值积分

- 常用调用方式：
  - q=quad('myFun',0,10);
  - q=quad(@myFun,0,10);
  - q=quad(myinlineFun,0,10);

  myFun: 函数文件定义的函数

  myinlineFun： 内联函数

  即：

  一般函数：使用单引号或@

  内联函数：直接使用函数名

6.2.2 数值积分

例：求定积分 $\int_0^{3\pi} \mathrm{e}^{-0.5x} \sin\left(\frac{\pi}{6} + x\right) \mathrm{d}x$

(1)建立被积函数

法 一：使用函数文件**fesin.m**。

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

法二：使用内联函数：

```
g=inline('exp(-0.5*x).*sin(x+pi/6)');
```

Note:

被积函数中，变量间的乘、除、乘方一定要用点运算！

6.2.2 数值积分

(2) 调用数值积分函数quad求定积分

```
S=quad('fesin',0,3*pi)
S=quad(@fesin,0,3*pi)
    %函数应采用上面两种形式
S=quad(g,0,3*pi)
   %内联函数只能用此形式
```

6.2.2 数值积分

● 梯形积分法

在科学实验和工程应用中，函数关系往往是不知道的，只有实验测定的一组样本点和样本值，这时，就无法使用quad等函数计算其定积分。此时可用梯形积分法：其输入参数是一组矢量

I=trapz(X,Y)

X:样本点，**Y**:样本值

```
x=0:0.01:pi;
z=trapz(x,sin(x));
```

z 是 sin(x) 从 0 到 pi 的积分

6.2.2 数值积分

例6.21：计算定积分 $I=\int_0^1 e^{-x^2}\mathrm{d}x$

梯形积分法：

```
X=0:0.01:1;
Y=exp(-X.^2);
 %生成函数关系数据向量
trapz(X,Y)
ans =
   0.7468
```

辛普生法：

```
f=inline('exp(-x.^2)');
        %内联函数
quad(f,0,1)
quadl(f,0,1)
ans =
   0.7468
```

- **Note:** trapz的值受样本量的影响！

6.2.2 数值积分

●多重定积分的数值求解

- **dblquad**：二重积分的数值解

  »dblquad(@fun,a,b,c,d,tol)

- **triplequad**：三重积分的数值解

  »triplequad(@fun,a,b,c,d,e,f,tol)

fun为被积函数，**[a，b]**为**x**的积分区域，**[c，d]**为**y**的积分区域，**[e，f]**为**z**的积分区域，参数**tol**的用法与函数**quad**完全相同。

6.2.2 数值积分

例6.22(p160)计算二重定积分 $I=\int_{-1}^{1}\int_{-2}^{2} e^{-x^2/2}\sin(x^2+y)\mathrm{d}x\mathrm{d}y$

（1）建立函数文件fxy.m：

```
function f=fxy(x,y)
f=exp(-x.^2/2).*sin(x.^2+y);
```

（2）调用dblquad函数求解。

```
I=dblquad(@fxy,-2,2,-1,1)
   I =  1.5745
```

思考：使用inline函数？

```
f=inline('exp(-x.^2/2).*sin(x.^2+y)','x','y');
I=dblquad(f,-2,2,-1,1)
   I =   1.5745
```

6.3 离散傅立叶变换

- 离散傅立叶变换 **(p160)**:

$$X(n)=\sum_{k=0}^{N-1}x(k)e^{-j(2\pi/N)nk} \qquad n=0,\ldots N-1$$

$$x(k)=\frac{1}{N}\sum_{n=0}^{N-1}X(n)e^{j(2\pi/N)nk} \qquad k=0,\ldots N-1$$

- 对**matlab**，下标从**1**开始，故写为

$$X(n)=\sum_{k=1}^{N}x(k)e^{-j(2\pi/N)nk} \qquad n=1,\ldots N$$

$$x(k)=\frac{1}{N}\sum_{n=1}^{N}X(n)e^{j(2\pi/N)nk} \qquad k=1,\ldots N$$

6.3 离散傅立叶变换

- **fft(X)**：返回向量**X**的离散傅立叶变换。
- **fft(X,N)**：计算**N**点离散傅立叶变换。

  限定向量的长度为**N**：

  若**X**的长度**<N**，则不足部分补上零；

  若**X**的长度**>N**，则删去超出**N**的那些元素。

  - 若**N**为**2**的幂次，则为以**2**为基数的快速傅立叶变换，否则为运算速度很慢的非**2**幂次的算法。
  - 当已知给出的样本数N_0不是**2**的幂次时，可以取一个**N**使它大于N_0且是**2**的幂次，利用**fft(X,N)**便可进行快速傅立叶变换。
- **fft(X,[],dim)**或**fft(X,N,dim)**：
- 逆变换：

  ifft(F)、**ifft(F,N)**

  ifft(F,[],dim)、**ifft(F,N,dim)**

6.3 离散傅立叶变换

例6-23 给定数学函数

$$x(t)=12\sin(2\pi\times 10t+\pi/4)+5\cos(2\pi\times 40t)$$

取N=128，试对t从0~1秒采样，用fft作快速傅立叶变换，绘制相应的振幅-频率图。

分析：在0~1秒时间范围内采样128点，从而可以确定采样周期和采样频率。

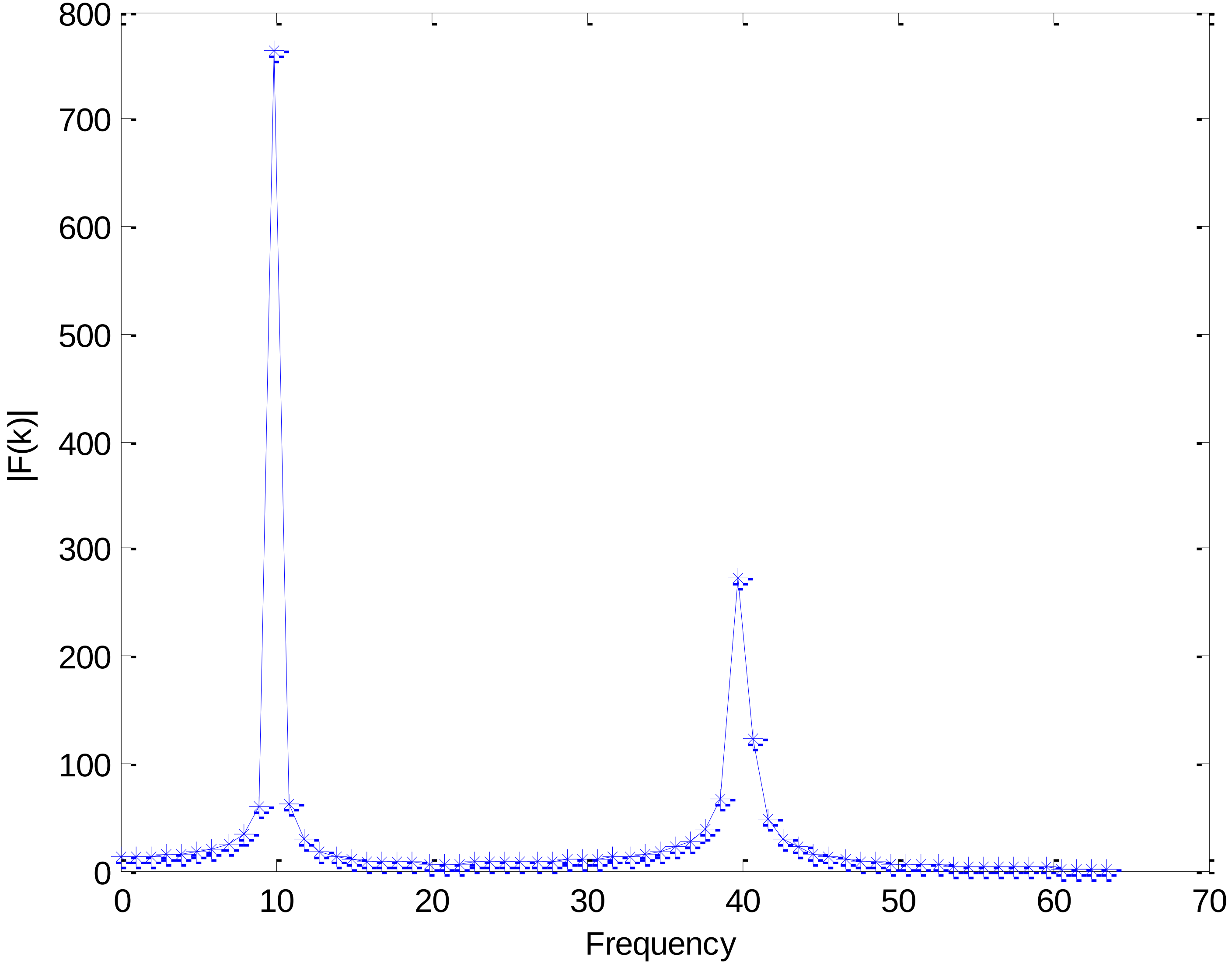
由于离散傅立叶变换时的下标应是从0到N-1，故在实际应用时下标应该前移1。

对离散傅立叶变换来说，其振幅| F(k)|是关于N/2对称的，故只须使k从0到N/2即可。

6.3 离散傅立叶变换

```
N=128;                    % 采样点数
T=1;                      % 采样时间终点
t=linspace(0,T,N);        % 给出N个采样时间ti(I=1:N)
x=12*sin(2*pi*10*t+pi/4)+5*cos(2*pi*40*t);
                          % 求各采样点样本值x
dt=t(2)-t(1);             % 采样周期
f=1/dt;                   % 采样频率(Hz)
X=fft(x);                 % 计算x的快速傅立叶变换X
F=X(1:N/2+1);             % F(k)=X(k)(k=1:N/2+1)
f=f*(0:N/2)/N;            % 使频率轴f从零开始
plot(f,abs(F),'-*')       % 绘制振幅-频率图
xlabel('Frequency');
ylabel('|F(k)|')
```

6.3 离散傅立叶变换



800
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
70
|F(k)|
Frequency

6.4.1 线性方程组直接求解

由包含n个未知数，n个方程组成的线性方程组如下：

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

将其写为矩阵形式： **Ax=b**

其中A为左端方程组系数构成的矩阵，b为右端的结果构成的矩阵，则其解为：

$$x = A^{-1}b$$

6.4.1 线性方程组直接求解

$x=A^{-1}b$在matlab中表示为： **x=inv(A)*b**

也可使用左除运算符： **x=A\b**

例6.24 用直接解法求解下列线性方程组。(p163)

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

```
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
```

6.4.1 线性方程组直接求解

例6.24 用直接解法求解下列线性方程组。(p163)

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

```
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
```

6.5.1 优化工具箱：非线性方程求解

- 许多现实问题需要求解方程f(x)=0
- **fzero** :求解任意函数的根
  - » fzero('myfun',1)   %一般函数
  - » fzero(@myfun,1)   %一般函数
  - » fzero(myinlinefun,1) %内联/匿名函数
- **e.g.:**
  - » myfun=inline('cos(exp(x))+x.^2-1');
  - » % myfun=@(x) cos(exp(x))+x.^2-1
  - » x=fzero(myfun,1)

6.5.2 优化工具箱：函数最小值

- **fminbnd**: 求函数在一个区间内的最小值点
  - » `[x,fmin]=fminbnd('myfun',-1,2);`
    - ➢ **x**: 函数**myfun**取得最小值的点
    - ➢ **vmin=myfun(x)**： 最小值
    - ➢ 函数必须是连续的
    - ➢ **x=fminbnd(‘-myfun’,-1,2)**
      ——myfun取最大值的点
- **fminsearch**:
  - » `[x,fmin]=fminsearch('myfun',0.5)`
    - ➢ 求**myfun**在**x=0.5**附近的极小值