## 温度识别

与16进制的转换相同

FFFF00 R在前,

R\*16\*16+G\*16+B

## 图像

### 图像上加网格

img= imread('temp.jpg');

title('xlfd')

M=2;

N=59;

imshow(img);

title('Íø¸ñ±ê¼ÇÍ¼Ïñ', 'FontWeight', 'Bold');

hold on;

[xt, yt] = meshgrid(round(linspace(1, size(img, 1), M)), ...

round(linspace(1, size(img, 2), N)));%Éú³ÉÊý¾Ýµã¾ØÕó

mesh(yt, xt, zeros(size(xt)), 'FaceColor', ...

'None', 'LineWidth', 1, ...

'EdgeColor', 'r');

### 三维图像(彩色图像)

data=imread('temp.jpg');

r=data(:,:,1);

g=data(:,:,2);

b=data(:,:,3);

分出r,g,b

### 读取灰度

a=imread(‘path’)这个如果是彩色的，则，读取出三维，要灰度的，得转

rgb2gray

### 两张图片相关系数

corr2(a,b)

两幅图像的相关系数求法：

img\_a = img\_a – MEAN(img\_a)

img\_b = img\_b – MEAN(img\_b)

Relation\_a\_b = r = sum(sum(img\_a.\*img\_b))/sqrt(sum(sum(img\_a.\*img\_a))\*sum(sum(img\_b.\*img\_b)));

matlab中corr2可以求解，代码如下：

a = a - mean2(a);   
b = b - mean2(b);   
r = sum(sum(a.\*b))/sqrt(sum(sum(a.\*a))\*sum(sum(b.\*b)));

a=imread('t1.TIF')

b=imread('t2.TIF')

a=rgb2gray(a);

b=rgb2gray(b);

a=im2double(a);

b=im2double(b);

a = a - mean2(a);

b = b - mean2(b);

r = sum(sum(a.\*b))/sqrt(sum(sum(a.\*a))\*sum(sum(b.\*b)))

与corr2(imread,imread)结果一致

### uint8转double

I2=**im2double**(I1)

## 快捷键

Ctrl +R注释

Ctrl+T反注释

## 基础

### 数字转字符串

num2str即可

### 字符串连接

在Matlab中，想要将两个字符串连接在一起，有以下的方法：

假定有两个字符串

>> str1='Iloveyou';str2='123';

**方法一**：用中括号将str1和str2像矩阵元素一样包含起来：

>> SC=[str1,str2]

SC =

Iloveyou123

（若想验证str1和str2确实被连接起来，可调用length函数测试SC的长度。）

**方法二**：用strcat函数

>> SB=strcat(str1,str2)

SB =

Iloveyou123

**注意**，strcat函数有许多用法，如下例：

>> strcat({'Red','Yellow'},{'Green','Blue'})

ans =

    'RedGreen'    'YellowBlue'

但下句则结果就不一样了：

>> strcat(['Red','Yellow'],['Green','Blue'])

ans =

RedYellowGreenBlue

**方法三**：利用sprintf函数

>> number=123;

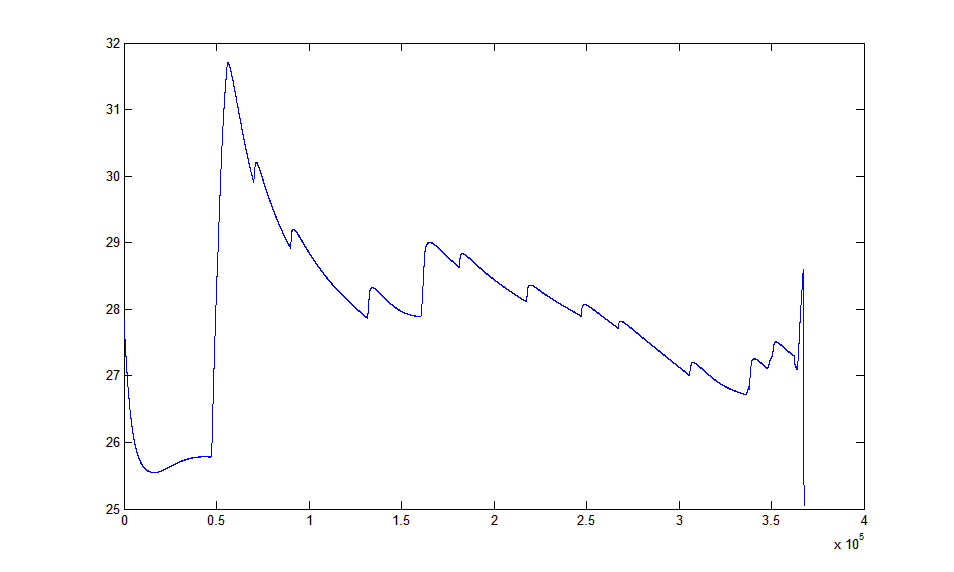
>> STR=sprintf('%s%d',str1,number)

STR =

Iloveyou123

利用class（STR）得到STR的类型为char。

### 读取txt



extv=importdata('D:\shuju\old1\extv.txt');

### 求向量的模

normest(f)

### 向量的逆序

fliplr(a)

### Find

test=magic(3);

test =

8 1 6

3 5 7

4 9 2

t=find(test>3)

t =

1

3

5

6

7

8

t表示索引,这个索引是竖着来的 8(1) 3(2) 4(3)…..理论等同imread得到的结果是行数\*列数(height\*width)

### 随机数

randint(1,1,[1,2]) 1-2的整数

## 支持向量机

### Libsvm安装

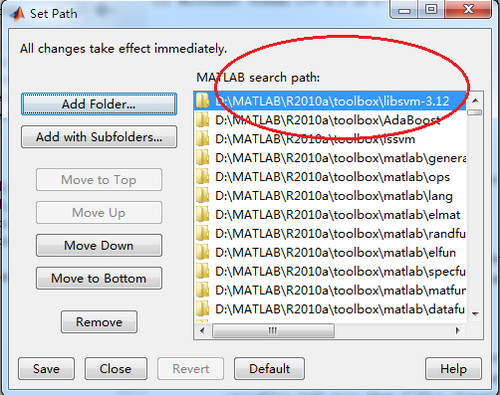
### MATLAB 安装使用libsvm详细步骤（附图）

2012-06-17 09:50:56|  分类： [IT](http://zjhello123.blog.163.com/blog/#m=0&t=1&c=fks_084070084084088075084085085095086083085074081087082064) |  标签：[matlab](http://zjhello123.blog.163.com/blog/#m=0&t=3&c=matlab)  [libsvm](http://zjhello123.blog.163.com/blog/#m=0&t=3&c=libsvm)  |举报|字号 订阅

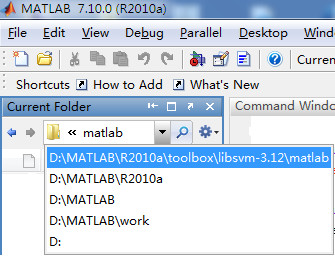
注：版本说明MATLAB R2010a,libsvm 3.12 ,VC++ 6.0  
  
////////////////////////////  
2013年4月12日，本博添加：

根据本文后面部分博友提出的在配置过程中出现的问题，其中需要特别强调的一点：整个过程，都是在 libsvm-3.12\matlab目录下操作的。如果这一点你忽视了，你不可能解决配置中报的Bug，即使重新安装matlab也不行。

本文的配置路径为 D:\MATLAB\R2010a\toolbox\libsvm-3.12\matlab，从始至终都是在这个目录下进行的。  
////////////////////////////  
详细步骤如下：  
1.下载libsvm  
http://www.csie.ntu.edu.tw/~cjlin/libsvm/  
我的matlab版本 R2010a，我的libsvm版本3.12  
  
  
2.解压至指定目录  
将libsvm解压至D:\MATLAB\R2010a\toolbox下，你也可以解压至你喜欢的地方。  
  
3.设置路径



下载下来的借口包里有svm的一些源文件，没有可执行的exe文件，所以，必须先将svmtrain等源文件编译为matlab可以使用的dll等文件。于是先选择编译器，如下：   
4.编译libsvm  
我的编译器是VC++ 6.0  
  
首先在matlab设置当前目录为 D:\MATLAB\R2010a\toolbox\libsvm-3.12\matlab



执行编译  
在matlab命令窗口中输入  
mex -setup  
显示

Please choose your compiler for building external interface (MEX) files:

Would you like mex to locate installed compilers [y]/n?  
  
输入y

Select a compiler:

[1] Lcc-win32 C 2.4.1 in D:\MATLAB\R2010a\sys\lcc

[2] Microsoft Visual C++ 2008 SP1 in D:\Program Files\Microsoft Visual Studio 9.0

[3] Microsoft Visual C++ 6.0 in D:\Program Files\Microsoft Visual Studio

[0] None   
  
Compiler:  
  
需要你选择编译器  
我输入的是3 即选择VC++ 6.0

Please verify your choices:

Compiler: Microsoft Visual C++ 6.0

Location: D:\Program Files\Microsoft Visual Studio

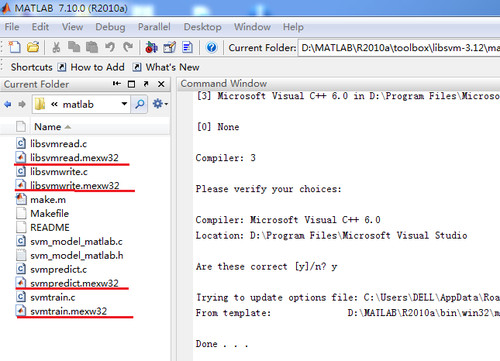
Are these correct [y]/n?  
再次确认自己的选择，输入y

Trying to update options file: C:\Users\DELL\AppData\Roaming\MathWorks\MATLAB\R2010a\mexopts.bat

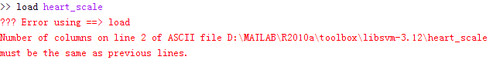
From template:              D:\MATLAB\R2010a\bin\win32\mexopts\msvc60opts.bat

Done . . . 

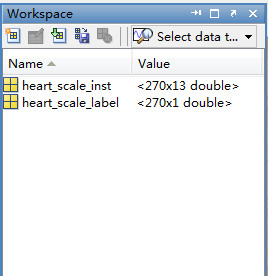
编译器选择好了之后，用make.m编译  
  
命令窗口输入make  
可以看到新编译出4个文件libsvmread.mexw32，libsvmwrite.mexw32，svmtrain.mexw32，svmpredict.mexw32

  
在命令窗口中输入

load heart\_scale;  
此时需注意，libsvm 3.12中提供的是c++版本的数据集heart\_scale，这里需要加载matlab版本的数据集。  
这两个数据集有什么不同呢？  
  
C++版本的数据集里面如果某一个样本的某一个特征为0，这个特征可以不写在数据集中，但是在matlab版本中，必须要写出来。  
否则的话，会报这样一个错误：

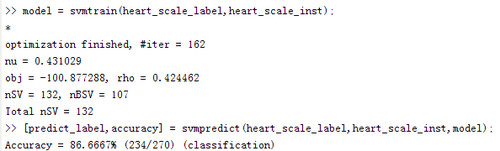


此时你可以选择下载matlab型的数据集，给一个链接。  
<http://download.csdn.net/detail/boruoshui/4881338>  
然后读取数据集  
libsvmread('heart\_scale');  
完成该步骤后发现Workspace中出现了heart\_scale\_inst 和 heart\_scale\_label,说明正确。



model=svmtrain(heart\_scale\_label,heart\_scale\_inst)

[predict\_lebel,accuracy]=svmpredict(heart\_scale\_label,heart\_scale\_inst,model)



证明libsvm安装成功，可以进行其他的实验了。

## 画图

### 基础

多图画到一个图里: 如果坐标单位相同的话，在画完第一个图后面加一条指令：hold on;就可以了！

散点图点的大小: plot(x,y,'.','markersize',8)%默认为6

颜色: bar(x,y,'g');

横坐标值:

set(gca,'xticklabel',{'04-01' '04-02' '04-08' '04-10' '04-15' '04-17' '04-20'});

显示不正常,要设置位置

x=[1,2,3];

set(gca,'xtick',x);

set(gca,'xticklabel',{'4ÔÂ15ÈÕ' '4ÔÂ17ÈÕ' '4ÔÂ20ÈÕ'});

* 例子

% dmy=['04-01' '04-02' '04-08' '04-10' '04-15' '04-17' '04-20']

% fw1=[290 310 316 318 293 290 305]

% fr1=[470 470 480 480 460 273 478]

% fw2=[310 310 318 309 309 303 311]

% fr2=[470 470 480 480 480 299 472]

% %plot(fw1,'color',[rand,rand,rand]);

% plot(fw1,'color','black');

% set(gca,'xticklabel',{'4ÔÂ1ÈÕ' '4ÔÂ2ÈÕ' '4ÔÂ8ÈÕ' '4ÔÂ10ÈÕ' '4ÔÂ15ÈÕ' '4ÔÂ17ÈÕ' '4ÔÂ20ÈÕ'});

% hold on;

% plot(fr1,'color','red');

% hold on;

% plot(fw2,'color','blue');

% hold on;

% plot(fr2,'color','green');

% legend('·þÎñÆ÷1Ð´ËÙ¶È','·þÎñÆ÷1¶ÁËÙ¶È','·þÎñÆ÷2Ð´ËÙ¶È','·þÎñÆ÷2¶ÁËÙ¶È')

% dks=['04-08' '04-10' '04-15' '04-17' '04-20']

% fw1=[323 333 308 321 312]

% fr1=[520 540 546 542 535]

% fw2=[327 323 308 317 318]

% fr2=[520 540 532 538 529]

% plot(fw1,'color','black');

% set(gca,'xticklabel',{ '4ÔÂ8ÈÕ' '4ÔÂ10ÈÕ' '4ÔÂ15ÈÕ' '4ÔÂ17ÈÕ' '4ÔÂ20ÈÕ'});

% hold on;

% plot(fr1,'color','red');

% hold on;

% plot(fw2,'color','blue');

% hold on;

% plot(fr2,'color','green');

% legend('·þÎñÆ÷1Ð´ËÙ¶È','·þÎñÆ÷1¶ÁËÙ¶È','·þÎñÆ÷2Ð´ËÙ¶È','·þÎñÆ÷2¶ÁËÙ¶È')

%

%

%

% dsy=['04-15' '04-17' '04-20']

% fw1=[308 312 306]

% fr1=[442 468 467]

% fw2=[324 320 311]

% fr2=[308 461 459]

%

%

%

% plot(fw1,'color','black');

%

% hold on;

% plot(fr1,'color','red');

% hold on;

% plot(fw2,'color','blue');

% hold on;

% plot(fr2,'color','green');

% x=[1,2,3];

% set(gca,'xtick',x);

% set(gca,'xticklabel',{'4ÔÂ15ÈÕ' '4ÔÂ17ÈÕ' '4ÔÂ20ÈÕ'});

% legend('·þÎñÆ÷1Ð´ËÙ¶È','·þÎñÆ÷1¶ÁËÙ¶È','·þÎñÆ÷2Ð´ËÙ¶È','·þÎñÆ÷2¶ÁËÙ¶È')

dbb=['04-13' '04-14' '04-15' '04-17']

fw7=[270 190 303 259]

fr7=[530 480 503 498]

fw6=[0 0 314 285]

fr6=[0 0 504 506]

plot(fw6,'color','black');

hold on;

plot(fr6,'color','red');

hold on;

plot(fw7,'color','blue');

hold on;

plot(fr7,'color','green');

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

set(gca,'xtick',x);

set(gca,'xticklabel',{'4ÔÂ13ÈÕ' '4ÔÂ14ÈÕ' '4ÔÂ15ÈÕ' '4ÔÂ17ÈÕ'});

legend('·þÎñÆ÷1Ð´ËÙ¶È','·þÎñÆ÷1¶ÁËÙ¶È','·þÎñÆ÷2Ð´ËÙ¶È','·þÎñÆ÷2¶ÁËÙ¶È')

设置图例:

legend('name1 ','name2 ')

#### 在原图基础上增加,不是重新画

hold on;

#### 直方图

matlab中函数bar绘制直方图中的应用函数bar(x)可以绘制直方图，这对统计或者数据采集非常直观实用。它共有四种形式：

bar,bar3,barh 和bar3h，其中bar 和bar3 分别用来绘制二维和三维竖直方图，barh 和b

ar3h 分

别用来绘制二维和三维水平直方图，调用格式是：

bar(x,y) 其中x 必须单调递增或递减，y 为n m× 矩阵，可视化结果为m 组，每

组n 个垂直柱，也就是把y 的行画在一起，同一列的数据用相同的颜色表示；

bar(x,y,width) （或bar(y,width)）指定每个直方条的宽度，如width>1，则直方条会重

叠，默认值为width=0.8；

bar(…,’grouped’) 使同一组直方条紧紧靠在一起；

bar(…,’stack’) 把同一组数据描述在一个直方条上。

例5.3.2

>> y=[5 3 2 9;4 7 2 7;1 5 7 3];

>> subplot(2,2,1),bar(y)

>> x=[5 9 11];

>> subplot(2,2,2),bar3(x,y)

>> subplot(2,2,3),bar(x,y,'grouped')

>> subplot(2,2,4),bar(rand(2,3),.75,'stack')

#### 三维散点图

x=[4229042.63 4230585.02 4231384.96 4231773.63 4233028.58 4233296.71 4235869.68 4236288.29];

y=[431695.4 441585.8 432745.6 436933.7 428734.4 431946.3 428705.0 432999.5];

z=[1.019 1.023 1.011 1.022 1.020 1.022 1.022 1.023];

scatter3(x,y,z)

#### 三维plot连线

互相连

设两点为：(x1,y1,z1),(x2,y2,z2)  
x=[x1,x2],y=[y1,y2],z=[z1,z2]  
plot3(x,y,z,'--');  
grid on  
axis（[xmin xmax ymin ymax zmin zmax]） ；%指定x坐标轴的尺度。

指定连(5,6,7),(2,8,1)

line([5,2],[6,8],[7,1]);

view(3);

#### 改变线条粗细

line([5,2],[6,8],[7,1],'linewidth',2);

view(3);也适用于plot等

#### 设置图的颜色rgb

plot([1,2],[3,4],'Color',[0,1,0])

#### 画箭头arrow3

调用,必须是这样用,始节点+单位向量

p1=[0,0,90];

p2=[61,17,100];

p2=p1+p2/normest(p2);

arrow3(p1,p2)

view(3)

grid on

第三方arrow3.m

function [h,yy,zz] = arrow(varargin)

% ARROW Draw a line with an arrowhead.

%

% ARROW(Start,Stop) draws a line with an arrow from Start to Stop (points

% should be vectors of length 2 or 3, or matrices with 2 or 3

% columns), and returns the graphics handle of the arrow(s).

%

% ARROW uses the mouse (click-drag) to create an arrow.

%

% ARROW DEMO & ARROW DEMO2 show 3-D & 2-D demos of the capabilities of ARROW.

%

% ARROW may be called with a normal argument list or a property-based list.

% ARROW(Start,Stop,Length,BaseAngle,TipAngle,Width,Page,CrossDir) is

% the full normal argument list, where all but the Start and Stop

% points are optional. If you need to specify a later argument (e.g.,

% Page) but want default values of earlier ones (e.g., TipAngle),

% pass an empty matrix for the earlier ones (e.g., TipAngle=[]).

%

% ARROW('Property1',PropVal1,'Property2',PropVal2,...) creates arrows with the

% given properties, using default values for any unspecified or given as

% 'default' or NaN. Some properties used for line and patch objects are

% used in a modified fashion, others are passed directly to LINE, PATCH,

% or SET. For a detailed properties explanation, call ARROW PROPERTIES.

%

% Start The starting points. B

% Stop The end points. /|\ ^

% Length Length of the arrowhead in pixels. /|||\ |

% BaseAngle Base angle in degrees (ADE). //|||\\ L|

% TipAngle Tip angle in degrees (ABC). ///|||\\\ e|

% Width Width of the base in pixels. ////|||\\\\ n|

% Page Use hardcopy proportions. /////|D|\\\\\ g|

% CrossDir Vector || to arrowhead plane. //// ||| \\\\ t|

% NormalDir Vector out of arrowhead plane. /// ||| \\\ h|

% Ends Which end has an arrowhead. //<----->|| \\ |

% ObjectHandles Vector of handles to update. / base ||| \ V

% E angle||<-------->C

% ARROW(H,'Prop1',PropVal1,...), where H is a |||tipangle

% vector of handles to previously-created arrows |||

% and/or line objects, will update the previously- |||

% created arrows according to the current view -->|A|<-- width

% and any specified properties, and will convert

% two-point line objects to corresponding arrows. ARROW(H) will update

% the arrows if the current view has changed. Root, figure, or axes

% handles included in H are replaced by all descendant Arrow objects.

%

% A property list can follow any specified normal argument list, e.g.,

% ARROW([1 2 3],[0 0 0],36,'BaseAngle',60) creates an arrow from (1,2,3) to

% the origin, with an arrowhead of length 36 pixels and 60-degree base angle.

%

% The basic arguments or properties can generally be vectorized to create

% multiple arrows with the same call. This is done by passing a property

% with one row per arrow, or, if all arrows are to have the same property

% value, just one row may be specified.

%

% You may want to execute AXIS(AXIS) before calling ARROW so it doesn't change

% the axes on you; ARROW determines the sizes of arrow components BEFORE the

% arrow is plotted, so if ARROW changes axis limits, arrows may be malformed.

%

% This version of ARROW uses features of MATLAB 5 and is incompatible with

% earlier MATLAB versions (ARROW for MATLAB 4.2c is available separately);

% some problems with perspective plots still exist.

% Copyright (c)1995-2002, Dr. Erik A. Johnson <JohnsonE@usc.edu>, 11/15/02

% Revision history:

% 11/15/02 EAJ Accomodate how MATLAB 6.5 handles NaN and logicals

% 7/28/02 EAJ Tried (but failed) work-around for MATLAB 6.x / OpenGL bug

% if zero 'Width' or not double-ended

% 11/10/99 EAJ Add logical() to eliminate zero index problem in MATLAB 5.3.

% 11/10/99 EAJ Corrected warning if axis limits changed on multiple axes.

% 11/10/99 EAJ Update e-mail address.

% 2/10/99 EAJ Some documentation updating.

% 2/24/98 EAJ Fixed bug if Start~=Stop but both colinear with viewpoint.

% 8/14/97 EAJ Added workaround for MATLAB 5.1 scalar logical transpose bug.

% 7/21/97 EAJ Fixed a few misc bugs.

% 7/14/97 EAJ Make arrow([],'Prop',...) do nothing (no old handles)

% 6/23/97 EAJ MATLAB 5 compatible version, release.

% 5/27/97 EAJ Added Line Arrows back in. Corrected a few bugs.

% 5/26/97 EAJ Changed missing Start/Stop to mouse-selected arrows.

% 5/19/97 EAJ MATLAB 5 compatible version, beta.

% 4/13/97 EAJ MATLAB 5 compatible version, alpha.

% 1/31/97 EAJ Fixed bug with multiple arrows and unspecified Z coords.

% 12/05/96 EAJ Fixed one more bug with log plots and NormalDir specified

% 10/24/96 EAJ Fixed bug with log plots and NormalDir specified

% 11/13/95 EAJ Corrected handling for 'reverse' axis directions

% 10/06/95 EAJ Corrected occasional conflict with SUBPLOT

% 4/24/95 EAJ A major rewrite.

% Fall 94 EAJ Original code.

% Things to be done:

% - segment parsing, computing, and plotting into separate subfunctions

% - change computing from Xform to Camera paradigms

% + this will help especially with 3-D perspective plots

% + if the WarpToFill section works right, remove warning code

% + when perpsective works properly, remove perspective warning code

% - add cell property values and struct property name/values (like get/set)

% - get rid of NaN as the "default" data label

% + perhaps change userdata to a struct and don't include (or leave

% empty) the values specified as default; or use a cell containing

% an empty matrix for a default value

% - add functionality of GET to retrieve current values of ARROW properties

% Many thanks to Keith Rogers <kerog@ai.mit.com> for his many excellent

% suggestions and beta testing. Check out his shareware package MATDRAW

% (at ftp://ftp.mathworks.com/pub/contrib/v5/graphics/matdraw/) -- he has

% permission to distribute ARROW with MATDRAW.

% Permission is granted to distribute ARROW with the toolboxes for the book

% "Solving Solid Mechanics Problems with MATLAB 5", by F. Golnaraghi et al.

% (Prentice Hall, 1999).

% global variable initialization

global ARROW\_PERSP\_WARN ARROW\_STRETCH\_WARN ARROW\_AXLIMITS

if isempty(ARROW\_PERSP\_WARN ), ARROW\_PERSP\_WARN =1; end;

if isempty(ARROW\_STRETCH\_WARN), ARROW\_STRETCH\_WARN=1; end;

% Handle callbacks

if (nargin>0 & isstr(varargin{1}) & strcmp(lower(varargin{1}),'callback')),

arrow\_callback(varargin{2:end}); return;

end;

% Are we doing the demo?

c = sprintf('\n');

if (nargin==1 & isstr(varargin{1})),

arg1 = lower(varargin{1});

if strncmp(arg1,'prop',4), arrow\_props;

elseif strncmp(arg1,'demo',4)

clf reset

demo\_info = arrow\_demo;

if ~strncmp(arg1,'demo2',5),

hh=arrow\_demo3(demo\_info);

else,

hh=arrow\_demo2(demo\_info);

end;

if (nargout>=1), h=hh; end;

elseif strncmp(arg1,'fixlimits',3),

arrow\_fixlimits(ARROW\_AXLIMITS);

ARROW\_AXLIMITS=[];

elseif strncmp(arg1,'help',4),

disp(help(mfilename));

else,

error([upper(mfilename) ' got an unknown single-argument string ''' deblank(arg1) '''.']);

end;

return;

end;

% Check # of arguments

if (nargout>3), error([upper(mfilename) ' produces at most 3 output arguments.']); end;

% find first property number

firstprop = nargin+1;

for k=1:length(varargin), if ~isnumeric(varargin{k}), firstprop=k; break; end; end;

lastnumeric = firstprop-1;

% check property list

if (firstprop<=nargin),

for k=firstprop:2:nargin,

curarg = varargin{k};

if ~isstr(curarg) | sum(size(curarg)>1)>1,

error([upper(mfilename) ' requires that a property name be a single string.']);

end;

end;

if (rem(nargin-firstprop,2)~=1),

error([upper(mfilename) ' requires that the property ''' ...

varargin{nargin} ''' be paired with a property value.']);

end;

end;

% default output

if (nargout>0), h=[]; end;

if (nargout>1), yy=[]; end;

if (nargout>2), zz=[]; end;

% set values to empty matrices

start = [];

stop = [];

len = [];

baseangle = [];

tipangle = [];

wid = [];

page = [];

crossdir = [];

ends = [];

ax = [];

oldh = [];

ispatch = [];

defstart = [NaN NaN NaN];

defstop = [NaN NaN NaN];

deflen = 16;

defbaseangle = 90;

deftipangle = 16;

defwid = 0;

defpage = 0;

defcrossdir = [NaN NaN NaN];

defends = 1;

defoldh = [];

defispatch = 1;

% The 'Tag' we'll put on our arrows

ArrowTag = 'Arrow';

% check for oldstyle arguments

if (firstprop==2),

% assume arg1 is a set of handles

oldh = varargin{1}(:);

if isempty(oldh), return; end;

elseif (firstprop>9),

error([upper(mfilename) ' takes at most 8 non-property arguments.']);

elseif (firstprop>2),

s = str2mat('start','stop','len','baseangle','tipangle','wid','page','crossdir');

for k=1:firstprop-1, eval([deblank(s(k,:)) '=varargin{k};']); end;

end;

% parse property pairs

extraprops={};

for k=firstprop:2:nargin,

prop = varargin{k};

val = varargin{k+1};

prop = [lower(prop(:)') ' '];

if strncmp(prop,'start' ,5), start = val;

elseif strncmp(prop,'stop' ,4), stop = val;

elseif strncmp(prop,'len' ,3), len = val(:);

elseif strncmp(prop,'base' ,4), baseangle = val(:);

elseif strncmp(prop,'tip' ,3), tipangle = val(:);

elseif strncmp(prop,'wid' ,3), wid = val(:);

elseif strncmp(prop,'page' ,4), page = val;

elseif strncmp(prop,'cross' ,5), crossdir = val;

elseif strncmp(prop,'norm' ,4), if (isstr(val)), crossdir=val; else, crossdir=val\*sqrt(-1); end;

elseif strncmp(prop,'end' ,3), ends = val;

elseif strncmp(prop,'object',6), oldh = val(:);

elseif strncmp(prop,'handle',6), oldh = val(:);

elseif strncmp(prop,'type' ,4), ispatch = val;

elseif strncmp(prop,'userd' ,5), %ignore it

else,

% make sure it is a valid patch or line property

eval('get(0,[''DefaultPatch'' varargin{k}]);err=0;','err=1;'); errstr=lasterr;

if (err), eval('get(0,[''DefaultLine'' varargin{k}]);err=0;','err=1;'); end;

if (err),

errstr(1:max(find(errstr==setstr(13)|errstr==setstr(10)))) = '';

error([upper(mfilename) ' got ' errstr]);

end;

extraprops={extraprops{:},varargin{k},val};

end;

end;

% Check if we got 'default' values

start = arrow\_defcheck(start ,defstart ,'Start' );

stop = arrow\_defcheck(stop ,defstop ,'Stop' );

len = arrow\_defcheck(len ,deflen ,'Length' );

baseangle = arrow\_defcheck(baseangle,defbaseangle,'BaseAngle' );

tipangle = arrow\_defcheck(tipangle ,deftipangle ,'TipAngle' );

wid = arrow\_defcheck(wid ,defwid ,'Width' );

crossdir = arrow\_defcheck(crossdir ,defcrossdir ,'CrossDir' );

page = arrow\_defcheck(page ,defpage ,'Page' );

ends = arrow\_defcheck(ends ,defends ,'' );

oldh = arrow\_defcheck(oldh ,[] ,'ObjectHandles');

ispatch = arrow\_defcheck(ispatch ,defispatch ,'' );

% check transpose on arguments

[m,n]=size(start ); if any(m==[2 3])&(n==1|n>3), start = start'; end;

[m,n]=size(stop ); if any(m==[2 3])&(n==1|n>3), stop = stop'; end;

[m,n]=size(crossdir); if any(m==[2 3])&(n==1|n>3), crossdir = crossdir'; end;

% convert strings to numbers

if ~isempty(ends) & isstr(ends),

endsorig = ends;

[m,n] = size(ends);

col = lower([ends(:,1:min(3,n)) ones(m,max(0,3-n))\*' ']);

ends = NaN\*ones(m,1);

oo = ones(1,m);

ii=find(all(col'==['non']'\*oo)'); if ~isempty(ii), ends(ii)=ones(length(ii),1)\*0; end;

ii=find(all(col'==['sto']'\*oo)'); if ~isempty(ii), ends(ii)=ones(length(ii),1)\*1; end;

ii=find(all(col'==['sta']'\*oo)'); if ~isempty(ii), ends(ii)=ones(length(ii),1)\*2; end;

ii=find(all(col'==['bot']'\*oo)'); if ~isempty(ii), ends(ii)=ones(length(ii),1)\*3; end;

if any(isnan(ends)),

ii = min(find(isnan(ends)));

error([upper(mfilename) ' does not recognize ''' deblank(endsorig(ii,:)) ''' as a valid ''Ends'' value.']);

end;

else,

ends = ends(:);

end;

if ~isempty(ispatch) & isstr(ispatch),

col = lower(ispatch(:,1));

patchchar='p'; linechar='l'; defchar=' ';

mask = col~=patchchar & col~=linechar & col~=defchar;

if any(mask),

error([upper(mfilename) ' does not recognize ''' deblank(ispatch(min(find(mask)),:)) ''' as a valid ''Type'' value.']);

end;

ispatch = (col==patchchar)\*1 + (col==linechar)\*0 + (col==defchar)\*defispatch;

else,

ispatch = ispatch(:);

end;

oldh = oldh(:);

% check object handles

if ~all(ishandle(oldh)), error([upper(mfilename) ' got invalid object handles.']); end;

% expand root, figure, and axes handles

if ~isempty(oldh),

ohtype = get(oldh,'Type');

mask = strcmp(ohtype,'root') | strcmp(ohtype,'figure') | strcmp(ohtype,'axes');

if any(mask),

oldh = num2cell(oldh);

for ii=find(mask)',

oldh(ii) = {findobj(oldh{ii},'Tag',ArrowTag)};

end;

oldh = cat(1,oldh{:});

if isempty(oldh), return; end; % no arrows to modify, so just leave

end;

end;

% largest argument length

[mstart,junk]=size(start); [mstop,junk]=size(stop); [mcrossdir,junk]=size(crossdir);

argsizes = [length(oldh) mstart mstop ...

length(len) length(baseangle) length(tipangle) ...

length(wid) length(page) mcrossdir length(ends) ];

args=['length(ObjectHandle) '; ...

'#rows(Start) '; ...

'#rows(Stop) '; ...

'length(Length) '; ...

'length(BaseAngle) '; ...

'length(TipAngle) '; ...

'length(Width) '; ...

'length(Page) '; ...

'#rows(CrossDir) '; ...

'#rows(Ends) '];

if (any(imag(crossdir(:))~=0)),

args(9,:) = '#rows(NormalDir) ';

end;

if isempty(oldh),

narrows = max(argsizes);

else,

narrows = length(oldh);

end;

if (narrows<=0), narrows=1; end;

% Check size of arguments

ii = find((argsizes~=0)&(argsizes~=1)&(argsizes~=narrows));

if ~isempty(ii),

s = args(ii',:);

while ((size(s,2)>1)&((abs(s(:,size(s,2)))==0)|(abs(s(:,size(s,2)))==abs(' ')))),

s = s(:,1:size(s,2)-1);

end;

s = [ones(length(ii),1)\*[upper(mfilename) ' requires that '] s ...

ones(length(ii),1)\*[' equal the # of arrows (' num2str(narrows) ').' c]];

s = s';

s = s(:)';

s = s(1:length(s)-1);

error(setstr(s));

end;

% check element length in Start, Stop, and CrossDir

if ~isempty(start),

[m,n] = size(start);

if (n==2),

start = [start NaN\*ones(m,1)];

elseif (n~=3),

error([upper(mfilename) ' requires 2- or 3-element Start points.']);

end;

end;

if ~isempty(stop),

[m,n] = size(stop);

if (n==2),

stop = [stop NaN\*ones(m,1)];

elseif (n~=3),

error([upper(mfilename) ' requires 2- or 3-element Stop points.']);

end;

end;

if ~isempty(crossdir),

[m,n] = size(crossdir);

if (n<3),

crossdir = [crossdir NaN\*ones(m,3-n)];

elseif (n~=3),

if (all(imag(crossdir(:))==0)),

error([upper(mfilename) ' requires 2- or 3-element CrossDir vectors.']);

else,

error([upper(mfilename) ' requires 2- or 3-element NormalDir vectors.']);

end;

end;

end;

% fill empty arguments

if isempty(start ), start = [Inf Inf Inf]; end;

if isempty(stop ), stop = [Inf Inf Inf]; end;

if isempty(len ), len = Inf; end;

if isempty(baseangle ), baseangle = Inf; end;

if isempty(tipangle ), tipangle = Inf; end;

if isempty(wid ), wid = Inf; end;

if isempty(page ), page = Inf; end;

if isempty(crossdir ), crossdir = [Inf Inf Inf]; end;

if isempty(ends ), ends = Inf; end;

if isempty(ispatch ), ispatch = Inf; end;

% expand single-column arguments

= ones(narrows,1);

if (size(start ,1)==1), start = o \* start ; end;

if (size(stop ,1)==1), stop = o \* stop ; end;

if (length(len )==1), len = o \* len ; end;

if (length(baseangle )==1), baseangle = o \* baseangle ; end;

if (length(tipangle )==1), tipangle = o \* tipangle ; end;

if (length(wid )==1), wid = o \* wid ; end;

if (length(page )==1), page = o \* page ; end;

if (size(crossdir ,1)==1), crossdir = o \* crossdir ; end;

if (length(ends )==1), ends = o \* ends ; end;

if (length(ispatch )==1), ispatch = o \* ispatch ; end;

ax = o \* gca;

% if we've got handles, get the defaults from the handles

if ~isempty(oldh),

for k=1:narrows,

oh = oldh(k);

ud = get(oh,'UserData');

ax(k) = get(oh,'Parent');

ohtype = get(oh,'Type');

if strcmp(get(oh,'Tag'),ArrowTag), % if it's an arrow already

if isinf(ispatch(k)), ispatch(k)=strcmp(ohtype,'patch'); end;

% arrow UserData format: [start' stop' len base tip wid page crossdir' ends]

start0 = ud(1:3);

stop0 = ud(4:6);

if (isinf(len(k))), len(k) = ud( 7); end;

if (isinf(baseangle(k))), baseangle(k) = ud( 8); end;

if (isinf(tipangle(k))), tipangle(k) = ud( 9); end;

if (isinf(wid(k))), wid(k) = ud(10); end;

if (isinf(page(k))), page(k) = ud(11); end;

if (isinf(crossdir(k,1))), crossdir(k,1) = ud(12); end;

if (isinf(crossdir(k,2))), crossdir(k,2) = ud(13); end;

if (isinf(crossdir(k,3))), crossdir(k,3) = ud(14); end;

if (isinf(ends(k))), ends(k) = ud(15); end;

elseif strcmp(ohtype,'line')|strcmp(ohtype,'patch'), % it's a non-arrow line or patch

convLineToPatch = 1; %set to make arrow patches when converting from lines.

if isinf(ispatch(k)), ispatch(k)=convLineToPatch|strcmp(ohtype,'patch'); end;

x=get(oh,'XData'); x=x(~isnan(x(:))); if isempty(x), x=NaN; end;

y=get(oh,'YData'); y=y(~isnan(y(:))); if isempty(y), y=NaN; end;

z=get(oh,'ZData'); z=z(~isnan(z(:))); if isempty(z), z=NaN; end;

start0 = [x(1) y(1) z(1) ];

stop0 = [x(end) y(end) z(end)];

else,

error([upper(mfilename) ' cannot convert ' ohtype ' objects.']);

end;

ii=find(isinf(start(k,:))); if ~isempty(ii), start(k,ii)=start0(ii); end;

ii=find(isinf(stop( k,:))); if ~isempty(ii), stop( k,ii)=stop0( ii); end;

end;

end;

% convert Inf's to NaN's

start( isinf(start )) = NaN;

stop( isinf(stop )) = NaN;

len( isinf(len )) = NaN;

baseangle( isinf(baseangle)) = NaN;

tipangle( isinf(tipangle )) = NaN;

wid( isinf(wid )) = NaN;

page( isinf(page )) = NaN;

crossdir( isinf(crossdir )) = NaN;

ends( isinf(ends )) = NaN;

ispatch( isinf(ispatch )) = NaN;

% set up the UserData data (here so not corrupted by log10's and such)

ud = [start stop len baseangle tipangle wid page crossdir ends];

% Set Page defaults

page = ~isnan(page) & trueornan(page);

% Get axes limits, range, min; correct for aspect ratio and log scale

axm = zeros(3,narrows);

axr = zeros(3,narrows);

axrev = zeros(3,narrows);

ap = zeros(2,narrows);

xyzlog = zeros(3,narrows);

limmin = zeros(2,narrows);

limrange = zeros(2,narrows);

oldaxlims = zeros(narrows,7);

oneax = all(ax==ax(1));

if (oneax),

T = zeros(4,4);

invT = zeros(4,4);

else,

T = zeros(16,narrows);

invT = zeros(16,narrows);

end;

axnotdone = logical(ones(size(ax)));

while (any(axnotdone)),

ii = min(find(axnotdone));

curax = ax(ii);

curpage = page(ii);

% get axes limits and aspect ratio

axl = [get(curax,'XLim'); get(curax,'YLim'); get(curax,'ZLim')];

oldaxlims(min(find(oldaxlims(:,1)==0)),:) = [curax reshape(axl',1,6)];

% get axes size in pixels (points)

u = get(curax,'Units');

axposoldunits = get(curax,'Position');

really\_curpage = curpage & strcmp(u,'normalized');

if (really\_curpage),

curfig = get(curax,'Parent');

pu = get(curfig,'PaperUnits');

set(curfig,'PaperUnits','points');

pp = get(curfig,'PaperPosition');

set(curfig,'PaperUnits',pu);

set(curax,'Units','pixels');

curapscreen = get(curax,'Position');

set(curax,'Units','normalized');

curap = pp.\*get(curax,'Position');

else,

set(curax,'Units','pixels');

curapscreen = get(curax,'Position');

curap = curapscreen;

end;

set(curax,'Units',u);

set(curax,'Position',axposoldunits);

% handle non-stretched axes position

str\_stretch = { 'DataAspectRatioMode' ; ...

'PlotBoxAspectRatioMode' ; ...

'CameraViewAngleMode' };

str\_camera = { 'CameraPositionMode' ; ...

'CameraTargetMode' ; ...

'CameraViewAngleMode' ; ...

'CameraUpVectorMode' };

notstretched = strcmp(get(curax,str\_stretch),'manual');

manualcamera = strcmp(get(curax,str\_camera),'manual');

if ~arrow\_WarpToFill(notstretched,manualcamera,curax),

% give a warning that this has not been thoroughly tested

if 0 & ARROW\_STRETCH\_WARN,

ARROW\_STRETCH\_WARN = 0;

strs = {str\_stretch{1:2},str\_camera{:}};

strs = [char(ones(length(strs),1)\*sprintf('\n ')) char(strs)]';

warning([upper(mfilename) ' may not yet work quite right ' ...

'if any of the following are ''manual'':' strs(:).']);

end;

% find the true pixel size of the actual axes

texttmp = text(axl(1,[1 2 2 1 1 2 2 1]), ...

axl(2,[1 1 2 2 1 1 2 2]), ...

axl(3,[1 1 1 1 2 2 2 2]),'');

set(texttmp,'Units','points');

textpos = get(texttmp,'Position');

delete(texttmp);

textpos = cat(1,textpos{:});

textpos = max(textpos(:,1:2)) - min(textpos(:,1:2));

% adjust the axes position

if (really\_curpage),

% adjust to printed size

textpos = textpos \* min(curap(3:4)./textpos);

curap = [curap(1:2)+(curap(3:4)-textpos)/2 textpos];

else,

% adjust for pixel roundoff

textpos = textpos \* min(curapscreen(3:4)./textpos);

curap = [curap(1:2)+(curap(3:4)-textpos)/2 textpos];

end;

end;

if ARROW\_PERSP\_WARN & ~strcmp(get(curax,'Projection'),'orthographic'),

ARROW\_PERSP\_WARN = 0;

warning([upper(mfilename) ' does not yet work right for 3-D perspective projection.']);

end;

% adjust limits for log scale on axes

curxyzlog = [strcmp(get(curax,'XScale'),'log'); ...

strcmp(get(curax,'YScale'),'log'); ...

strcmp(get(curax,'ZScale'),'log')];

if (any(curxyzlog)),

ii = find([curxyzlog;curxyzlog]);

if (any(axl(ii)<=0)),

error([upper(mfilename) ' does not support non-positive limits on log-scaled axes.']);

else,

axl(ii) = log10(axl(ii));

end;

end;

% correct for 'reverse' direction on axes;

curreverse = [strcmp(get(curax,'XDir'),'reverse'); ...

strcmp(get(curax,'YDir'),'reverse'); ...

strcmp(get(curax,'ZDir'),'reverse')];

ii = find(curreverse);

if ~isempty(ii),

axl(ii,[1 2])=-axl(ii,[2 1]);

end;

% compute the range of 2-D values

curT = get(curax,'Xform');

lim = curT\*[0 1 0 1 0 1 0 1;0 0 1 1 0 0 1 1;0 0 0 0 1 1 1 1;1 1 1 1 1 1 1 1];

lim = lim(1:2,:)./([1;1]\*lim(4,:));

curlimmin = min(lim')';

curlimrange = max(lim')' - curlimmin;

curinvT = inv(curT);

if (~oneax),

curT = curT.';

curinvT = curinvT.';

curT = curT(:);

curinvT = curinvT(:);

end;

% check which arrows to which cur corresponds

ii = find((ax==curax)&(page==curpage));

oo = ones(1,length(ii));

axr(:,ii) = diff(axl')' \* oo;

axm(:,ii) = axl(:,1) \* oo;

axrev(:,ii) = curreverse \* oo;

ap(:,ii) = curap(3:4)' \* oo;

xyzlog(:,ii) = curxyzlog \* oo;

limmin(:,ii) = curlimmin \* oo;

limrange(:,ii) = curlimrange \* oo;

if (oneax),

T = curT;

invT = curinvT;

else,

T(:,ii) = curT \* oo;

invT(:,ii) = curinvT \* oo;

end;

axnotdone(ii) = zeros(1,length(ii));

end;

oldaxlims(oldaxlims(:,1)==0,:)=[];

% correct for log scales

curxyzlog = xyzlog.';

ii = find(curxyzlog(:));

if ~isempty(ii),

start( ii) = real(log10(start( ii)));

stop( ii) = real(log10(stop( ii)));

if (all(imag(crossdir)==0)), % pulled (ii) subscript on crossdir, 12/5/96 eaj

crossdir(ii) = real(log10(crossdir(ii)));

end;

end;

% correct for reverse directions

ii = find(axrev.');

if ~isempty(ii),

start( ii) = -start( ii);

stop( ii) = -stop( ii);

crossdir(ii) = -crossdir(ii);

end;

% transpose start/stop values

start = start.';

stop = stop.';

% take care of defaults, page was done above

ii=find(isnan(start(:) )); if ~isempty(ii), start(ii) = axm(ii)+axr(ii)/2; end;

ii=find(isnan(stop(:) )); if ~isempty(ii), stop(ii) = axm(ii)+axr(ii)/2; end;

ii=find(isnan(crossdir(:) )); if ~isempty(ii), crossdir(ii) = zeros(length(ii),1); end;

ii=find(isnan(len )); if ~isempty(ii), len(ii) = ones(length(ii),1)\*deflen; end;

ii=find(isnan(baseangle )); if ~isempty(ii), baseangle(ii) = ones(length(ii),1)\*defbaseangle; end;

ii=find(isnan(tipangle )); if ~isempty(ii), tipangle(ii) = ones(length(ii),1)\*deftipangle; end;

ii=find(isnan(wid )); if ~isempty(ii), wid(ii) = ones(length(ii),1)\*defwid; end;

ii=find(isnan(ends )); if ~isempty(ii), ends(ii) = ones(length(ii),1)\*defends; end;

% transpose rest of values

len = len.';

baseangle = baseangle.';

tipangle = tipangle.';

wid = wid.';

page = page.';

crossdir = crossdir.';

ends = ends.';

ax = ax.';

% given x, a 3xN matrix of points in 3-space;

% want to convert to X, the corresponding 4xN 2-space matrix

%

% tmp1=[(x-axm)./axr; ones(1,size(x,1))];

% if (oneax), X=T\*tmp1;

% else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=T.\*tmp1;

% tmp2=zeros(4,4\*N); tmp2(:)=tmp1(:);

% X=zeros(4,N); X(:)=sum(tmp2)'; end;

% X = X ./ (ones(4,1)\*X(4,:));

% for all points with start==stop, start=stop-(verysmallvalue)\*(up-direction);

ii = find(all(start==stop));

if ~isempty(ii),

% find an arrowdir vertical on screen and perpendicular to viewer

% transform to 2-D

tmp1 = [(stop(:,ii)-axm(:,ii))./axr(:,ii);ones(1,length(ii))];

if (oneax), twoD=T\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=T(:,ii).\*tmp1;

tmp2=zeros(4,4\*length(ii)); tmp2(:)=tmp1(:);

twoD=zeros(4,length(ii)); twoD(:)=sum(tmp2)'; end;

twoD=twoD./(ones(4,1)\*twoD(4,:));

% move the start point down just slightly

tmp1 = twoD + [0;-1/1000;0;0]\*(limrange(2,ii)./ap(2,ii));

% transform back to 3-D

if (oneax), threeD=invT\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=invT(:,ii).\*tmp1;

tmp2=zeros(4,4\*length(ii)); tmp2(:)=tmp1(:);

threeD=zeros(4,length(ii)); threeD(:)=sum(tmp2)'; end;

start(:,ii) = (threeD(1:3,:)./(ones(3,1)\*threeD(4,:))).\*axr(:,ii)+axm(:,ii);

end;

% compute along-arrow points

% transform Start points

tmp1=[(start-axm)./axr;ones(1,narrows)];

if (oneax), X0=T\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=T.\*tmp1;

tmp2=zeros(4,4\*narrows); tmp2(:)=tmp1(:);

X0=zeros(4,narrows); X0(:)=sum(tmp2)'; end;

X0=X0./(ones(4,1)\*X0(4,:));

% transform Stop points

tmp1=[(stop-axm)./axr;ones(1,narrows)];

if (oneax), Xf=T\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=T.\*tmp1;

tmp2=zeros(4,4\*narrows); tmp2(:)=tmp1(:);

Xf=zeros(4,narrows); Xf(:)=sum(tmp2)'; end;

Xf=Xf./(ones(4,1)\*Xf(4,:));

% compute pixel distance between points

D = sqrt(sum(((Xf(1:2,:)-X0(1:2,:)).\*(ap./limrange)).^2));

D = D + (D==0); %eaj new 2/24/98

% compute and modify along-arrow distances

len1 = len;

len2 = len - (len.\*tan(tipangle/180\*pi)-wid/2).\*tan((90-baseangle)/180\*pi);

slen0 = zeros(1,narrows);

slen1 = len1 .\* ((ends==2)|(ends==3));

slen2 = len2 .\* ((ends==2)|(ends==3));

len0 = zeros(1,narrows);

len1 = len1 .\* ((ends==1)|(ends==3));

len2 = len2 .\* ((ends==1)|(ends==3));

% for no start arrowhead

ii=find((ends==1)&(D<len2));

if ~isempty(ii),

slen0(ii) = D(ii)-len2(ii);

end;

% for no end arrowhead

ii=find((ends==2)&(D<slen2));

if ~isempty(ii),

len0(ii) = D(ii)-slen2(ii);

end;

len1 = len1 + len0;

len2 = len2 + len0;

slen1 = slen1 + slen0;

slen2 = slen2 + slen0;

% note: the division by D below will probably not be accurate if both

% of the following are true:

% 1. the ratio of the line length to the arrowhead

% length is large

% 2. the view is highly perspective.

% compute stoppoints

tmp1=X0.\*(ones(4,1)\*(len0./D))+Xf.\*(ones(4,1)\*(1-len0./D));

if (oneax), tmp3=invT\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=invT.\*tmp1;

tmp2=zeros(4,4\*narrows); tmp2(:)=tmp1(:);

tmp3=zeros(4,narrows); tmp3(:)=sum(tmp2)'; end;

stoppoint = tmp3(1:3,:)./(ones(3,1)\*tmp3(4,:)).\*axr+axm;

% compute tippoints

tmp1=X0.\*(ones(4,1)\*(len1./D))+Xf.\*(ones(4,1)\*(1-len1./D));

if (oneax), tmp3=invT\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=invT.\*tmp1;

tmp2=zeros(4,4\*narrows); tmp2(:)=tmp1(:);

tmp3=zeros(4,narrows); tmp3(:)=sum(tmp2)'; end;

tippoint = tmp3(1:3,:)./(ones(3,1)\*tmp3(4,:)).\*axr+axm;

% compute basepoints

tmp1=X0.\*(ones(4,1)\*(len2./D))+Xf.\*(ones(4,1)\*(1-len2./D));

if (oneax), tmp3=invT\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=invT.\*tmp1;

tmp2=zeros(4,4\*narrows); tmp2(:)=tmp1(:);

tmp3=zeros(4,narrows); tmp3(:)=sum(tmp2)'; end;

basepoint = tmp3(1:3,:)./(ones(3,1)\*tmp3(4,:)).\*axr+axm;

% compute startpoints

tmp1=X0.\*(ones(4,1)\*(1-slen0./D))+Xf.\*(ones(4,1)\*(slen0./D));

if (oneax), tmp3=invT\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=invT.\*tmp1;

tmp2=zeros(4,4\*narrows); tmp2(:)=tmp1(:);

tmp3=zeros(4,narrows); tmp3(:)=sum(tmp2)'; end;

startpoint = tmp3(1:3,:)./(ones(3,1)\*tmp3(4,:)).\*axr+axm;

% compute stippoints

tmp1=X0.\*(ones(4,1)\*(1-slen1./D))+Xf.\*(ones(4,1)\*(slen1./D));

if (oneax), tmp3=invT\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=invT.\*tmp1;

tmp2=zeros(4,4\*narrows); tmp2(:)=tmp1(:);

tmp3=zeros(4,narrows); tmp3(:)=sum(tmp2)'; end;

stippoint = tmp3(1:3,:)./(ones(3,1)\*tmp3(4,:)).\*axr+axm;

% compute sbasepoints

tmp1=X0.\*(ones(4,1)\*(1-slen2./D))+Xf.\*(ones(4,1)\*(slen2./D));

if (oneax), tmp3=invT\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=invT.\*tmp1;

tmp2=zeros(4,4\*narrows); tmp2(:)=tmp1(:);

tmp3=zeros(4,narrows); tmp3(:)=sum(tmp2)'; end;

sbasepoint = tmp3(1:3,:)./(ones(3,1)\*tmp3(4,:)).\*axr+axm;

% compute cross-arrow directions for arrows with NormalDir specified

if (any(imag(crossdir(:))~=0)),

ii = find(any(imag(crossdir)~=0));

crossdir(:,ii) = cross((stop(:,ii)-start(:,ii))./axr(:,ii), ...

imag(crossdir(:,ii))).\*axr(:,ii);

end;

% compute cross-arrow directions

basecross = crossdir + basepoint;

tipcross = crossdir + tippoint;

sbasecross = crossdir + sbasepoint;

stipcross = crossdir + stippoint;

ii = find(all(crossdir==0)|any(isnan(crossdir)));

if ~isempty(ii),

numii = length(ii);

% transform start points

tmp1 = [basepoint(:,ii) tippoint(:,ii) sbasepoint(:,ii) stippoint(:,ii)];

tmp1 = (tmp1-axm(:,[ii ii ii ii])) ./ axr(:,[ii ii ii ii]);

tmp1 = [tmp1; ones(1,4\*numii)];

if (oneax), X0=T\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=T(:,[ii ii ii ii]).\*tmp1;

tmp2=zeros(4,16\*numii); tmp2(:)=tmp1(:);

X0=zeros(4,4\*numii); X0(:)=sum(tmp2)'; end;

X0=X0./(ones(4,1)\*X0(4,:));

% transform stop points

tmp1 = [(2\*stop(:,ii)-start(:,ii)-axm(:,ii))./axr(:,ii);ones(1,numii)];

tmp1 = [tmp1 tmp1 tmp1 tmp1];

if (oneax), Xf=T\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=T(:,[ii ii ii ii]).\*tmp1;

tmp2=zeros(4,16\*numii); tmp2(:)=tmp1(:);

Xf=zeros(4,4\*numii); Xf(:)=sum(tmp2)'; end;

Xf=Xf./(ones(4,1)\*Xf(4,:));

% compute perpendicular directions

pixfact = ((limrange(1,ii)./limrange(2,ii)).\*(ap(2,ii)./ap(1,ii))).^2;

pixfact = [pixfact pixfact pixfact pixfact];

pixfact = [pixfact;1./pixfact];

[dummyval,jj] = max(abs(Xf(1:2,:)-X0(1:2,:)));

jj1 = ((1:4)'\*ones(1,length(jj))==ones(4,1)\*jj);

jj2 = ((1:4)'\*ones(1,length(jj))==ones(4,1)\*(3-jj));

jj3 = jj1(1:2,:);

Xf(jj1)=Xf(jj1)+(Xf(jj1)-X0(jj1)==0); %eaj new 2/24/98

Xp = X0;

Xp(jj2) = X0(jj2) + ones(sum(jj2(:)),1);

Xp(jj1) = X0(jj1) - (Xf(jj2)-X0(jj2))./(Xf(jj1)-X0(jj1)) .\* pixfact(jj3);

% inverse transform the cross points

if (oneax), Xp=invT\*Xp;

else, tmp1=[Xp;Xp;Xp;Xp]; tmp1=invT(:,[ii ii ii ii]).\*tmp1;

tmp2=zeros(4,16\*numii); tmp2(:)=tmp1(:);

Xp=zeros(4,4\*numii); Xp(:)=sum(tmp2)'; end;

Xp=(Xp(1:3,:)./(ones(3,1)\*Xp(4,:))).\*axr(:,[ii ii ii ii])+axm(:,[ii ii ii ii]);

basecross(:,ii) = Xp(:,0\*numii+(1:numii));

tipcross(:,ii) = Xp(:,1\*numii+(1:numii));

sbasecross(:,ii) = Xp(:,2\*numii+(1:numii));

stipcross(:,ii) = Xp(:,3\*numii+(1:numii));

end;

% compute all points

% compute start points

axm11 = [axm axm axm axm axm axm axm axm axm axm axm];

axr11 = [axr axr axr axr axr axr axr axr axr axr axr];

st = [stoppoint tippoint basepoint sbasepoint stippoint startpoint stippoint sbasepoint basepoint tippoint stoppoint];

tmp1 = (st - axm11) ./ axr11;

tmp1 = [tmp1; ones(1,size(tmp1,2))];

if (oneax), X0=T\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=[T T T T T T T T T T T].\*tmp1;

tmp2=zeros(4,44\*narrows); tmp2(:)=tmp1(:);

X0=zeros(4,11\*narrows); X0(:)=sum(tmp2)'; end;

X0=X0./(ones(4,1)\*X0(4,:));

% compute stop points

tmp1 = ([start tipcross basecross sbasecross stipcross stop stipcross sbasecross basecross tipcross start] ...

axm11) ./ axr11;

tmp1 = [tmp1; ones(1,size(tmp1,2))];

if (oneax), Xf=T\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=[T T T T T T T T T T T].\*tmp1;

tmp2=zeros(4,44\*narrows); tmp2(:)=tmp1(:);

Xf=zeros(4,11\*narrows); Xf(:)=sum(tmp2)'; end;

Xf=Xf./(ones(4,1)\*Xf(4,:));

% compute lengths

len0 = len.\*((ends==1)|(ends==3)).\*tan(tipangle/180\*pi);

slen0 = len.\*((ends==2)|(ends==3)).\*tan(tipangle/180\*pi);

le = [zeros(1,narrows) len0 wid/2 wid/2 slen0 zeros(1,narrows) -slen0 -wid/2 -wid/2 -len0 zeros(1,narrows)];

aprange = ap./limrange;

aprange = [aprange aprange aprange aprange aprange aprange aprange aprange aprange aprange aprange];

D = sqrt(sum(((Xf(1:2,:)-X0(1:2,:)).\*aprange).^2));

Dii=find(D==0); if ~isempty(Dii), D=D+(D==0); le(Dii)=zeros(1,length(Dii)); end; %should fix DivideByZero warnings

tmp1 = X0.\*(ones(4,1)\*(1-le./D)) + Xf.\*(ones(4,1)\*(le./D));

% inverse transform

if (oneax), tmp3=invT\*tmp1;

else, tmp1=[tmp1;tmp1;tmp1;tmp1]; tmp1=[invT invT invT invT invT invT invT invT invT invT invT].\*tmp1;

tmp2=zeros(4,44\*narrows); tmp2(:)=tmp1(:);

tmp3=zeros(4,11\*narrows); tmp3(:)=sum(tmp2)'; end;

pts = tmp3(1:3,:)./(ones(3,1)\*tmp3(4,:)) .\* axr11 + axm11;

% correct for ones where the crossdir was specified

ii = find(~(all(crossdir==0)|any(isnan(crossdir))));

if ~isempty(ii),

D1 = [pts(:,1\*narrows+ii)-pts(:,9\*narrows+ii) ...

pts(:,2\*narrows+ii)-pts(:,8\*narrows+ii) ...

pts(:,3\*narrows+ii)-pts(:,7\*narrows+ii) ...

pts(:,4\*narrows+ii)-pts(:,6\*narrows+ii) ...

pts(:,6\*narrows+ii)-pts(:,4\*narrows+ii) ...

pts(:,7\*narrows+ii)-pts(:,3\*narrows+ii) ...

pts(:,8\*narrows+ii)-pts(:,2\*narrows+ii) ...

pts(:,9\*narrows+ii)-pts(:,1\*narrows+ii)]/2;

ii = ii'\*ones(1,8) + ones(length(ii),1)\*[1:4 6:9]\*narrows;

ii = ii(:)';

pts(:,ii) = st(:,ii) + D1;

end;

% readjust for reverse directions

iicols=(1:narrows)'; iicols=iicols(:,ones(1,11)); iicols=iicols(:).';

tmp1=axrev(:,iicols);

ii = find(tmp1(:)); if ~isempty(ii), pts(ii)=-pts(ii); end;

% readjust for log scale on axes

tmp1=xyzlog(:,iicols);

ii = find(tmp1(:)); if ~isempty(ii), pts(ii)=10.^pts(ii); end;

% compute the x,y,z coordinates of the patches;

ii = narrows\*(0:10)'\*ones(1,narrows) + ones(11,1)\*(1:narrows);

ii = ii(:)';

x = zeros(11,narrows);

y = zeros(11,narrows);

z = zeros(11,narrows);

x(:) = pts(1,ii)';

y(:) = pts(2,ii)';

z(:) = pts(3,ii)';

% do the output

if (nargout<=1),

% % create or modify the patches

newpatch = trueornan(ispatch) & (isempty(oldh)|~strcmp(get(oldh,'Type'),'patch'));

newline = ~trueornan(ispatch) & (isempty(oldh)|~strcmp(get(oldh,'Type'),'line'));

if isempty(oldh), H=zeros(narrows,1); else, H=oldh; end;

% % make or modify the arrows

for k=1:narrows,

if all(isnan(ud(k,[3 6])))&arrow\_is2DXY(ax(k)), zz=[]; else, zz=z(:,k); end;

% work around a MATLAB 6.x OpenGL bug -- 7/28/02

xx=x(:,k); yy=y(:,k);

mask=any([ones(1,2+size(zz,2));diff([xx yy zz],[],1)],2);

xx=xx(mask); yy=yy(mask); if ~isempty(zz), zz=zz(mask); end;

% plot the patch or line

xyz = {'XData',xx,'YData',yy,'ZData',zz,'Tag',ArrowTag};

if newpatch(k)|newline(k),

if newpatch(k),

H(k) = patch(xyz{:});

else,

H(k) = line(xyz{:});

end;

if ~isempty(oldh), arrow\_copyprops(oldh(k),H(k)); end;

else,

if ispatch(k), xyz={xyz{:},'CData',[]}; end;

set(H(k),xyz{:});

end;

end;

if ~isempty(oldh), delete(oldh(oldh~=H)); end;

% % additional properties

set(H,'Clipping','off');

set(H,{'UserData'},num2cell(ud,2));

if (length(extraprops)>0), set(H,extraprops{:}); end;

% handle choosing arrow Start and/or Stop locations if unspecified

[H,oldaxlims,errstr] = arrow\_clicks(H,ud,x,y,z,ax,oldaxlims);

if ~isempty(errstr), error([upper(mfilename) ' got ' errstr]); end;

% set the output

if (nargout>0), h=H; end;

% make sure the axis limits did not change

if isempty(oldaxlims),

ARROW\_AXLIMITS = [];

else,

lims = get(oldaxlims(:,1),{'XLim','YLim','ZLim'})';

lims = reshape(cat(2,lims{:}),6,size(lims,2));

mask = arrow\_is2DXY(oldaxlims(:,1));

oldaxlims(mask,6:7) = lims(5:6,mask)';

ARROW\_AXLIMITS = oldaxlims(find(any(oldaxlims(:,2:7)'~=lims)),:);

if ~isempty(ARROW\_AXLIMITS),

warning(arrow\_warnlimits(ARROW\_AXLIMITS,narrows));

end;

end;

else,

% don't create the patch, just return the data

h=x;

yy=y;

zz=z;

end;

function out = arrow\_defcheck(in,def,prop)

% check if we got 'default' values

out = in;

if ~isstr(in), return; end;

if size(in,1)==1 & strncmp(lower(in),'def',3),

out = def;

elseif ~isempty(prop),

error([upper(mfilename) ' does not recognize ''' in(:)' ''' as a valid ''' prop ''' string.']);

end;

function [H,oldaxlims,errstr] = arrow\_clicks(H,ud,x,y,z,ax,oldaxlims)

% handle choosing arrow Start and/or Stop locations if necessary

errstr = '';

if isempty(H)|isempty(ud)|isempty(x), return; end;

% determine which (if any) need Start and/or Stop

needStart = all(isnan(ud(:,1:3)'))';

needStop = all(isnan(ud(:,4:6)'))';

mask = any(needStart|needStop);

if ~any(mask), return; end;

ud(~mask,:)=[]; ax(:,~mask)=[];

x(:,~mask)=[]; y(:,~mask)=[]; z(:,~mask)=[];

% make them invisible for the time being

set(H,'Visible','off');

% save the current axes and limits modes; set to manual for the time being

oldAx = gca;

limModes=get(ax(:),{'XLimMode','YLimMode','ZLimMode'});

set(ax(:),{'XLimMode','YLimMode','ZLimMode'},{'manual','manual','manual'});

% loop over each arrow that requires attention

jj = find(mask);

for ii=1:length(jj),

h = H(jj(ii));

axes(ax(ii));

% figure out correct call

if needStart(ii), prop='Start'; else, prop='Stop'; end;

[wasInterrupted,errstr] = arrow\_click(needStart(ii)&needStop(ii),h,prop,ax(ii));

% handle errors and control-C

if wasInterrupted,

delete(H(jj(ii:end)));

H(jj(ii:end))=[];

oldaxlims(jj(ii:end),:)=[];

break;

end;

end;

% restore the axes and limit modes

axes(oldAx);

set(ax(:),{'XLimMode','YLimMode','ZLimMode'},limModes);

function [wasInterrupted,errstr] = arrow\_click(lockStart,H,prop,ax)

% handle the clicks for one arrow

fig = get(ax,'Parent');

% save some things

oldFigProps = {'Pointer','WindowButtonMotionFcn','WindowButtonUpFcn'};

oldFigValue = get(fig,oldFigProps);

oldArrowProps = {'EraseMode'};

oldArrowValue = get(H,oldArrowProps);

set(H,'EraseMode','background'); %because 'xor' makes shaft invisible unless Width>1

global ARROW\_CLICK\_H ARROW\_CLICK\_PROP ARROW\_CLICK\_AX ARROW\_CLICK\_USE\_Z

ARROW\_CLICK\_H=H; ARROW\_CLICK\_PROP=prop; ARROW\_CLICK\_AX=ax;

ARROW\_CLICK\_USE\_Z=~arrow\_is2DXY(ax)|~arrow\_planarkids(ax);

set(fig,'Pointer','crosshair');

% set up the WindowButtonMotion so we can see the arrow while moving around

set(fig,'WindowButtonUpFcn','set(gcf,''WindowButtonUpFcn'','''')', ...

'WindowButtonMotionFcn','');

if ~lockStart,

set(H,'Visible','on');

set(fig,'WindowButtonMotionFcn',[mfilename '(''callback'',''motion'');']);

end;

% wait for the button to be pressed

[wasKeyPress,wasInterrupted,errstr] = arrow\_wfbdown(fig);

% if we wanted to click-drag, set the Start point

if lockStart & ~wasInterrupted,

pt = arrow\_point(ARROW\_CLICK\_AX,ARROW\_CLICK\_USE\_Z);

feval(mfilename,H,'Start',pt,'Stop',pt);

set(H,'Visible','on');

ARROW\_CLICK\_PROP='Stop';

set(fig,'WindowButtonMotionFcn',[mfilename '(''callback'',''motion'');']);

% wait for the mouse button to be released

eval('waitfor(fig,''WindowButtonUpFcn'','''');','wasInterrupted=1;');

if wasInterrupted, errstr=lasterr; end;

end;

if ~wasInterrupted, feval(mfilename,'callback','motion'); end;

% restore some things

set(gcf,oldFigProps,oldFigValue);

set(H,oldArrowProps,oldArrowValue);

function arrow\_callback(varargin)

% handle redrawing callbacks

if nargin==0, return; end;

str = varargin{1};

if ~isstr(str), error([upper(mfilename) ' got an invalid Callback command.']); end;

s = lower(str);

if strcmp(s,'motion'),

% motion callback

global ARROW\_CLICK\_H ARROW\_CLICK\_PROP ARROW\_CLICK\_AX ARROW\_CLICK\_USE\_Z

feval(mfilename,ARROW\_CLICK\_H,ARROW\_CLICK\_PROP,arrow\_point(ARROW\_CLICK\_AX,ARROW\_CLICK\_USE\_Z));

drawnow;

else,

error([upper(mfilename) ' does not recognize ''' str(:).' ''' as a valid Callback option.']);

end;

function out = arrow\_point(ax,use\_z)

% return the point on the given axes

if nargin==0, ax=gca; end;

if nargin<2, use\_z=~arrow\_is2DXY(ax)|~arrow\_planarkids(ax); end;

out = get(ax,'CurrentPoint');

out = out(1,:);

if ~use\_z, out=out(1:2); end;

function [wasKeyPress,wasInterrupted,errstr] = arrow\_wfbdown(fig)

% wait for button down ignoring object ButtonDownFcn's

if nargin==0, fig=gcf; end;

errstr = '';

% save ButtonDownFcn values

objs = findobj(fig);

buttonDownFcns = get(objs,'ButtonDownFcn');

mask=~strcmp(buttonDownFcns,''); objs=objs(mask); buttonDownFcns=buttonDownFcns(mask);

set(objs,'ButtonDownFcn','');

% save other figure values

figProps = {'KeyPressFcn','WindowButtonDownFcn'};

figValue = get(fig,figProps);

% do the real work

set(fig,'KeyPressFcn','set(gcf,''KeyPressFcn'','''',''WindowButtonDownFcn'','''');', ...

'WindowButtonDownFcn','set(gcf,''WindowButtonDownFcn'','''')');

lasterr('');

wasInterrupted=0; eval('waitfor(fig,''WindowButtonDownFcn'','''');','wasInterrupted=1;');

wasKeyPress = ~wasInterrupted & strcmp(get(fig,'KeyPressFcn'),'');

if wasInterrupted, errstr=lasterr; end;

% restore ButtonDownFcn and other figure values

set(objs,'ButtonDownFcn',buttonDownFcns);

set(fig,figProps,figValue);

function [out,is2D] = arrow\_is2DXY(ax)

% check if axes are 2-D X-Y plots

% may not work for modified camera angles, etc.

out = logical(zeros(size(ax))); % 2-D X-Y plots

is2D = out; % any 2-D plots

views = get(ax(:),{'View'});

views = cat(1,views{:});

out(:) = abs(views(:,2))==90;

is2D(:) = out(:) | all(rem(views',90)==0)';

function out = arrow\_planarkids(ax)

% check if axes descendents all have empty ZData (lines,patches,surfaces)

out = logical(ones(size(ax)));

allkids = get(ax(:),{'Children'});

for k=1:length(allkids),

kids = get([findobj(allkids{k},'flat','Type','line')

findobj(allkids{k},'flat','Type','patch')

findobj(allkids{k},'flat','Type','surface')],{'ZData'});

for j=1:length(kids),

if ~isempty(kids{j}), out(k)=logical(0); break; end;

end;

end;

function arrow\_fixlimits(axlimits)

% reset the axis limits as necessary

if isempty(axlimits), disp([upper(mfilename) ' does not remember any axis limits to reset.']); end;

for k=1:size(axlimits,1),

if any(get(axlimits(k,1),'XLim')~=axlimits(k,2:3)), set(axlimits(k,1),'XLim',axlimits(k,2:3)); end;

if any(get(axlimits(k,1),'YLim')~=axlimits(k,4:5)), set(axlimits(k,1),'YLim',axlimits(k,4:5)); end;

if any(get(axlimits(k,1),'ZLim')~=axlimits(k,6:7)), set(axlimits(k,1),'ZLim',axlimits(k,6:7)); end;

end;

function out = arrow\_WarpToFill(notstretched,manualcamera,curax)

% check if we are in "WarpToFill" mode.

out = strcmp(get(curax,'WarpToFill'),'on');

% 'WarpToFill' is undocumented, so may need to replace this by

% out = ~( any(notstretched) & any(manualcamera) );

function out = arrow\_warnlimits(axlimits,narrows)

% create a warning message if we've changed the axis limits

msg = '';

switch (size(axlimits,1))

case 1, msg='';

case 2, msg='on two axes ';

otherwise, msg='on several axes ';

end;

msg = [upper(mfilename) ' changed the axis limits ' msg ...

'when adding the arrow'];

if (narrows>1), msg=[msg 's']; end;

out = [msg '.' sprintf('\n') ' Call ' upper(mfilename) ...

' FIXLIMITS to reset them now.'];

function arrow\_copyprops(fm,to)

% copy line properties to patches

props = {'EraseMode','LineStyle','LineWidth','Marker','MarkerSize',...

'MarkerEdgeColor','MarkerFaceColor','ButtonDownFcn', ...

'Clipping','DeleteFcn','BusyAction','HandleVisibility', ...

'Selected','SelectionHighlight','Visible'};

lineprops = {'Color', props{:}};

patchprops = {'EdgeColor',props{:}};

patch2props = {'FaceColor',patchprops{:}};

fmpatch = strcmp(get(fm,'Type'),'patch');

topatch = strcmp(get(to,'Type'),'patch');

set(to( fmpatch& topatch),patch2props,get(fm( fmpatch& topatch),patch2props)); %p->p

set(to(~fmpatch&~topatch),lineprops, get(fm(~fmpatch&~topatch),lineprops )); %l->l

set(to( fmpatch&~topatch),lineprops, get(fm( fmpatch&~topatch),patchprops )); %p->l

set(to(~fmpatch& topatch),patchprops, get(fm(~fmpatch& topatch),lineprops) ,'FaceColor','none'); %l->p

function arrow\_props

% display further help info about ARROW properties

c = sprintf('\n');

disp([c ...

'ARROW Properties: Default values are given in [square brackets], and other' c ...

' acceptable equivalent property names are in (parenthesis).' c c ...

' Start The starting points. For N arrows, B' c ...

' this should be a Nx2 or Nx3 matrix. /|\ ^' c ...

' Stop The end points. For N arrows, this /|||\ |' c ...

' should be a Nx2 or Nx3 matrix. //|||\\ L|' c ...

' Length Length of the arrowhead (in pixels on ///|||\\\ e|' c ...

' screen, points on a page). [16] (Len) ////|||\\\\ n|' c ...

' BaseAngle Angle (degrees) of the base angle /////|D|\\\\\ g|' c ...

' ADE. For a simple stick arrow, use //// ||| \\\\ t|' c ...

' BaseAngle=TipAngle. [90] (Base) /// ||| \\\ h|' c ...

' TipAngle Angle (degrees) of tip angle ABC. //<----->|| \\ |' c ...

' [16] (Tip) / base ||| \ V' c ...

' Width Width of the base in pixels. Not E angle ||<-------->C' c ...

' the ''LineWidth'' prop. [0] (Wid) |||tipangle' c ...

' Page If provided, non-empty, and not NaN, |||' c ...

' this causes ARROW to use hardcopy |||' c ...

' rather than onscreen proportions. A' c ...

' This is important if screen aspect --> <-- width' c ...

' ratio and hardcopy aspect ratio are ----CrossDir---->' c ...

' vastly different. []' c...

' CrossDir A vector giving the direction towards which the fletches' c ...

' on the arrow should go. [computed such that it is perpen-' c ...

' dicular to both the arrow direction and the view direction' c ...

' (i.e., as if it was pasted on a normal 2-D graph)] (Note' c ...

' that CrossDir is a vector. Also note that if an axis is' c ...

' plotted on a log scale, then the corresponding component' c ...

' of CrossDir must also be set appropriately, i.e., to 1 for' c ...

' no change in that direction, >1 for a positive change, >0' c ...

' and <1 for negative change.)' c ...

' NormalDir A vector normal to the fletch direction (CrossDir is then' c ...

' computed by the vector cross product [Line]x[NormalDir]). []' c ...

' (Note that NormalDir is a vector. Unlike CrossDir,' c ...

' NormalDir is used as is regardless of log-scaled axes.)' c ...

' Ends Set which end has an arrowhead. Valid values are ''none'',' c ...

' ''stop'', ''start'', and ''both''. [''stop''] (End)' c...

' ObjectHandles Vector of handles to previously-created arrows to be' c ...

' updated or line objects to be converted to arrows.' c ...

' [] (Object,Handle)' c ]);

function out = arrow\_demo

% demo

% create the data

[x,y,z] = peaks;

[ddd,out.iii]=max(z(:));

out.axlim = [min(x(:)) max(x(:)) min(y(:)) max(y(:)) min(z(:)) max(z(:))];

% modify it by inserting some NaN's

[m,n] = size(z);

m = floor(m/2);

n = floor(n/2);

z(1:m,1:n) = NaN\*ones(m,n);

% graph it

clf('reset');

out.hs=surf(x,y,z);

out.x=x; out.y=y; out.z=z;

xlabel('x'); ylabel('y');

function h = arrow\_demo3(in)

% set the view

axlim = in.axlim;

axis(axlim);

zlabel('z');

%set(in.hs,'FaceColor','interp');

view(viewmtx(-37.5,30,20));

title(['Demo of the capabilities of the ARROW function in 3-D']);

% Normal blue arrow

h1 = feval(mfilename,[axlim(1) axlim(4) 4],[-.8 1.2 4], ...

'EdgeColor','b','FaceColor','b');

% Normal white arrow, clipped by the surface

h2 = feval(mfilename,axlim([1 4 6]),[0 2 4]);

t=text(-2.4,2.7,7.7,'arrow clipped by surf');

% Baseangle<90

h3 = feval(mfilename,[3 .125 3.5],[1.375 0.125 3.5],30,50);

t2=text(3.1,.125,3.5,'local maximum');

% Baseangle<90, fill and edge colors different

h4 = feval(mfilename,axlim(1:2:5)\*.5,[0 0 0],36,60,25, ...

'EdgeColor','b','FaceColor','c');

t3=text(axlim(1)\*.5,axlim(3)\*.5,axlim(5)\*.5-.75,'origin');

set(t3,'HorizontalAlignment','center');

% Baseangle>90, black fill

h5 = feval(mfilename,[-2.9 2.9 3],[-1.3 .4 3.2],30,120,[],6, ...

'EdgeColor','r','FaceColor','k','LineWidth',2);

% Baseangle>90, no fill

h6 = feval(mfilename,[-2.9 2.9 1.3],[-1.3 .4 1.5],30,120,[],6, ...

'EdgeColor','r','FaceColor','none','LineWidth',2);

% Stick arrow

h7 = feval(mfilename,[-1.6 -1.65 -6.5],[0 -1.65 -6.5],[],16,16);

t4=text(-1.5,-1.65,-7.25,'global mininum');

set(t4,'HorizontalAlignment','center');

% Normal, black fill

h8 = feval(mfilename,[-1.4 0 -7.2],[-1.4 0 -3],'FaceColor','k');

t5=text(-1.5,0,-7.75,'local minimum');

set(t5,'HorizontalAlignment','center');

% Gray fill, crossdir specified, 'LineStyle' --

h9 = feval(mfilename,[-3 2.2 -6],[-3 2.2 -.05],36,[],27,6,[],[0 -1 0], ...

'EdgeColor','k','FaceColor',.75\*[1 1 1],'LineStyle','--');

% a series of normal arrows, linearly spaced, crossdir specified

h10y=(0:4)'/3;

h10 = feval(mfilename,[-3\*ones(size(h10y)) h10y -6.5\*ones(size(h10y))], ...

[-3\*ones(size(h10y)) h10y -.05\*ones(size(h10y))], ...

12,[],[],[],[],[0 -1 0]);

% a series of normal arrows, linearly spaced

h11x=(1:.33:2.8)';

h11 = feval(mfilename,[h11x -3\*ones(size(h11x)) 6.5\*ones(size(h11x))], ...

[h11x -3\*ones(size(h11x)) -.05\*ones(size(h11x))]);

% series of magenta arrows, radially oriented, crossdir specified

h12x=2; h12y=-3; h12z=axlim(5)/2; h12xr=1; h12zr=h12z; ir=.15;or=.81;

h12t=(0:11)'/6\*pi;

h12 = feval(mfilename, ...

[h12x+h12xr\*cos(h12t)\*ir h12y\*ones(size(h12t)) ...

h12z+h12zr\*sin(h12t)\*ir],[h12x+h12xr\*cos(h12t)\*or ...

h12y\*ones(size(h12t)) h12z+h12zr\*sin(h12t)\*or], ...

10,[],[],[],[], ...

[-h12xr\*sin(h12t) zeros(size(h12t)) h12zr\*cos(h12t)],...

'FaceColor','none','EdgeColor','m');

% series of normal arrows, tangentially oriented, crossdir specified

or13=.91; h13t=(0:.5:12)'/6\*pi;

locs = [h12x+h12xr\*cos(h13t)\*or13 h12y\*ones(size(h13t)) h12z+h12zr\*sin(h13t)\*or13];

h13 = feval(mfilename,locs(1:end-1,:),locs(2:end,:),6);

% arrow with no line ==> oriented downwards

h14 = feval(mfilename,[3 3 .100001],[3 3 .1],30);

t6=text(3,3,3.6,'no line'); set(t6,'HorizontalAlignment','center');

% arrow with arrowheads at both ends

h15 = feval(mfilename,[-.5 -3 -3],[1 -3 -3],'Ends','both','FaceColor','g', ...

'Length',20,'Width',3,'CrossDir',[0 0 1],'TipAngle',25);

h=[h1;h2;h3;h4;h5;h6;h7;h8;h9;h10;h11;h12;h13;h14;h15];

function h = arrow\_demo2(in)

axlim = in.axlim;

dolog = 1;

if (dolog), set(in.hs,'YData',10.^get(in.hs,'YData')); end;

shading('interp');

view(2);

title(['Demo of the capabilities of the ARROW function in 2-D']);

hold on; [C,H]=contour(in.x,in.y,in.z,20,'-'); hold off;

for k=H',

set(k,'ZData',(axlim(6)+1)\*ones(size(get(k,'XData'))),'Color','k');

if (dolog), set(k,'YData',10.^get(k,'YData')); end;

end;

if (dolog), axis([axlim(1:2) 10.^axlim(3:4)]); set(gca,'YScale','log');

else, axis(axlim(1:4)); end;

% Normal blue arrow

start = [axlim(1) axlim(4) axlim(6)+2];

stop = [in.x(in.iii) in.y(in.iii) axlim(6)+2];

if (dolog), start(:,2)=10.^start(:,2); stop(:,2)=10.^stop(:,2); end;

h1 = feval(mfilename,start,stop,'EdgeColor','b','FaceColor','b');

% three arrows with varying fill, width, and baseangle

start = [-3 -3 10; -3 -1.5 10; -1.5 -3 10];

stop = [-.03 -.03 10; -.03 -1.5 10; -1.5 -.03 10];

if (dolog), start(:,2)=10.^start(:,2); stop(:,2)=10.^stop(:,2); end;

h2 = feval(mfilename,start,stop,24,[90;60;120],[],[0;0;4],'Ends',str2mat('both','stop','stop'));

set(h2(2),'EdgeColor',[0 .35 0],'FaceColor',[0 .85 .85]);

set(h2(3),'EdgeColor','r','FaceColor',[1 .5 1]);

h=[h1;h2];

function out = trueornan(x)

if isempty(x),

out=x;

else,

out = isnan(x);

out(~out) = x(~out);

end;

#### 画箭头quiver3

最直接的用法放六个值　　quiver3(x,y,z,u,v,w)  
  
例如quiver3(1,2,3,4,5,6) 就以点(1,2,3)为起点作一个(4,5,6)向量,　即在(1,2,3)一个指向(5,7,9)的箭头.  
  
x,y,z,u,v,w必须长度一样,　就可以作出一个三维的向量图.　向量由(u,v,w)决定,　所在位置由(x,y,z)决定.

p1=[0,40,90];

p2=[61,17,100];

c=p2-p1;

quiver3(p1(1),p1(2),p1(3),c(1),c(2),c(3))

view(3)

grid on

#### 绕y轴旋转一圈得到三维图形

clear;clc

y=0:0.001:1;

x=y.^0.5;

[X,Y,Z]=cylinder(x,20);

mesh(X,Y,Z)

#### 画圆柱

R=1;%半径

a=2;%原点x

b=3;%原点y

h=4;%圆柱高度

m=100;%分割线条数

[x,y,z]=cylinder(R,m);%(0,0)为圆心,高度为[0,1],半径为1的圆柱

x=x+a;%平移x轴

y=y+b;%平移y轴

z=h\*z;%高度放大h倍

mesh(x,y,z)%重新绘图

#### 图形翻转

clear;clc

AA=imread('fire.bmp');

imshow(AA);

for k=1:3

BB(:,:,k)=flipud(AA(:,:,k));%上下翻转

B(:,:,k)=fliplr(AA(:,:,k));%水平翻转

end

figure;

imshow(BB);

figure

imshow(B);

## 多元一次线性回归

clear

clc

data = [5 5 5 5 5 5 5 10 10 10 10 10 10 10 15 15 15 15 15 15 15 20 20 20 20 20 20 20 25 25 25 25 25 25 25 30 30 30 30 30 30 30

144 176 208 240 272 320 512 144 176 208 240 272 320 512 144 176 208 240 272 320 512 144 176 208 240 272 320 512 144 176 208 240 272 320 512 144 176 208 240 272 320 512

135.346 138.193 147.369 137.871 140.782 143.022 146.985 179.075 230.787 236.99 240.18 232.734 248.473 249.812 218.834 242.56 276.038 282.54 293.582 320.17 279.204 215.68 286.505 313.645 334.782 325.526 332.145 337.282 231.54 261.884 324.214 343.436 367.167 372.446 340.181 259.584 302.132 313.351 347.63 340.478 414.03 443.249

]';

x1 = data(:,1);

x2 = data(:,2);

y = data(:,3);

X = [ones(size(x1)) x1.\*x1 x1 x2.\*x2 x2 x1.\*x2];

[b] = regress(y,X)

scatter3(x1,x2,y,'filled')

hold on

x1fit = min(x1):0.5:max(x1);

x2fit = min(x2):0.5:max(x2);

[X1FIT,X2FIT] = meshgrid(x1fit,x2fit);

YFIT = b(1)+ b(2)\*X1FIT.\*X1FIT+b(3)\*X1FIT + b(4)\*X2FIT.\*X2FIT + b(5)\*X2FIT + b(6)\*X1FIT.\*X2FIT;

mesh(X1FIT,X2FIT,YFIT)

xlabel('x1')

ylabel('x2')

zlabel('Y')

view(140,30)

## 经验累积分布函数(求<x)那个F(x)

[fi,xi] = ecdf(x);

fx = spline(xi(2:end),fi(2:end),x);

[fi,yi] = ecdf(y);

fy = spline(yi(2:end),fi(2:end),y);

scatter(fx,fy);

求经验分布函数的时候是要去除多余重复值的，可以通过插值方法计算原始数据所对应的经验分布函数值,

如果不用spline,则求出来的fx,fy长度不一样,不能画图,因为去掉了重复

spline(x,y,xi);   
这个是根据己知的x，y数据，用样条函数插值出xi处的值。即由x,y的值计算出xi对应的函数值

## 求积分

一重定积分(数值积分)

clear;clc  
df=@(x)(cos(pi\*cos(x))+1).^2./sin(x)  
quad(df,0,pi)



也可以syms x;syms y;不用引号

**q = dblquad(fun,xmin,xmax,ymin,ymax,tol,method)**  
矩形区域二重数值积分，一般区域二重积分参见NIT(数值积分工具箱)的quad2dggen函数。

例9 计算下面二重积分  
>>F = @(x,y)y\*sin(x)+x\*cos(y);  
>Q = dblquad(F,pi,2\*pi,0,pi)

Q =

-9.8696

是这样的 quad2dggen函数的第一重积分的上下限必须是函数，即使数据我们也要表示成函数的形式  
  
因为quad2dggen函数内部是使用feval来计算积分值的，而feval的参数必须是句柄或是句柄字符串  
如下所示  
fun=@(x,y)normpdf(x,10,1).\*normpdf(y,5,1);  
low=@(y)0;  
up=@(y)10;  
Q=quad2dggen(fun,low,up,-10,0)

Matlab2012a貌似不支持

先下载nit积分工具箱,然后:

fun=@(s,t)( ( (1+s.\*s-2\*p\*s\*t+t\*t)/(v\*(1-p\*p)) ).^(-v/2-1) )/(2\*pi\*(1-p\*p)^0.5);

low=@(y)0;

up=@(y)1;

Q=quad2dggen(fun,low,up,0,1)

PS:

乘方和乘自己的时候

用.的,不然出错

无穷积分

syms t s;

s=1/(1+t^2);

res=int(s,-inf,0)

数值定积分[f=@(x)x.\*x](mailto:f=@(x)x.*x)

quadgk(f,0,1)

## 求导

matlab求导命令diff调用格式:  
  
diff(函数) ， 求的[一阶导数](http://zhidao.baidu.com/search?word=%E4%B8%80%E9%98%B6%E5%AF%BC%E6%95%B0&fr=qb_search_exp&ie=utf8)；  
  
diff(函数， n) ， 求的n阶[导数](http://zhidao.baidu.com/search?word=%E5%AF%BC%E6%95%B0&fr=qb_search_exp&ie=utf8)(n是具体整数)；  
  
diff(函数，变量名)， 求对的[偏导数](http://zhidao.baidu.com/search?word=%E5%81%8F%E5%AF%BC%E6%95%B0&fr=qb_search_exp&ie=utf8)；  
  
diff(函数， 变量名，n) ，求对的n阶[偏导数](http://zhidao.baidu.com/search?word=%E5%81%8F%E5%AF%BC%E6%95%B0&fr=qb_search_exp&ie=utf8)；  
  
matlab求[雅可比矩阵](http://zhidao.baidu.com/search?word=%E9%9B%85%E5%8F%AF%E6%AF%94%E7%9F%A9%E9%98%B5&fr=qb_search_exp&ie=utf8)命令jacobian，调用格式:  
  
jacobian([函数；函数； 函数]， [])给出矩阵:   
  
另外   
解[微分方程](http://zhidao.baidu.com/search?word=%E5%BE%AE%E5%88%86%E6%96%B9%E7%A8%8B&fr=qb_search_exp&ie=utf8)可以用desolve  
例  
>> x=solve('x^2=y','x')  
   
x =  
   
 y^(1/2)  
 -y^(1/2)

## syms符号变量赋值

syms x1 x2  
f=2\*x1^2-2\*x1\*x2+x2^2+2\*x1-2\*x2;   
g=[diff(f,x1); diff(f,x2)];  
x1=0;x2=0;  
subs(g) % 加这句

元素替换的意思subs(f,old,new);

subs(x\*y,{x,y},{[0 1;-1 0],[1 -1;-2 1]})

## 求函数的逆函数值

syms x y; %定义变量

y=x\*x   
y=finverse(y,x) %求逆

## matlab坐标轴设置

matlab坐标轴设置 (2011-05-06 11:07:36)转载▼

标签： 杂谈 分类： Matlab应用

a=linspace(1,2,10)

plot(a,'--pr','linewidth',1.5,'MarkerEdgeColor','r','MarkerFaceColor','m','MarkerSize',10)

legend('a','Location','best')

title('a','FontName','Times New Roman','FontWeight','Bold','FontSize',16)

xlabel('T','FontName','Times New Roman','FontSize',14)

ylabel('a','FontName','Times New Roman','FontSize',14,'Rotation',0)

axis auto equal

set(gca,'FontName','Times New Roman','FontSize',14)

曲线线型、颜色和标记点类型

plot(X1,Y1,LineSpec, …) 通过字符串LineSpec指定曲线的线型、颜色及数据点的标记类型。

线型 颜色 数据点标记类型

标识符 意义 标识符意义 标识符 意义

- 实线 r 红色 + 加号

-. 点划线 g 绿色 o 圆圈

-- 虚线 b 蓝色 \* 星号

: 点线 c 蓝绿色 . 点

m 洋红色 x 交叉符号

y 黄色 square(或s) 方格

k 黑色 diamond(或d) 菱形

w 白色 ^ 向上的三角形

v 向下的三角形

向左的三角形

< 向右的三角形

pentagram(或p) 五边形

hexagram(或h) 六边形

设置曲线线宽、标记点大小，标记点边框颜色和标记点填充颜色等。

plot(…,’Property Name’, Property Value, …)

Property Name 意义 选项

LineWidth 线宽 数值，如0.5，1等，单位为points

MarkerEdgeColor 标记点边框线条颜色颜色字符，如’g’, ’b’等

MarkerFaceColor 标记点内部区域填充颜色颜色字符

MarkerSize 标记点大小 数值，单位为points

坐标轴设置

范围设置：

axis([xmin xmax ymin ymax])设置坐标轴在指定的区间

axis auto 将当前绘图区的坐标轴范围设置为MATLAB自动调整的区间

axis manual 冻结当前坐标轴范围，以后叠加绘图都在当前坐标轴范围内显示

axis tight 采用紧密模式设置当前坐标轴范围，即以用户数据范围为坐标轴范围比例：

axis equal 等比例坐标轴

axis square 以当前坐标轴范围为基础，将坐标轴区域调整为方格形

axis normal 自动调整纵横轴比例，使当前坐标轴范围内的图形显示达到最佳效果

范围选项和比例设置可以联合使用，默认的设置为axis auto normal

坐标轴刻度设置

set(gca, ’XTick’, [0 1 2]) X坐标轴刻度数据点位置

set(gca,'XTickLabel',{'a','b','c'}) X坐标轴刻度处显示的字符

set(gca,'FontName','Times New Roman','FontSize',14)设置坐标轴刻度字体名称，大小

‘FontWeight’,’bold’ 加粗 ‘FontAngle’,’italic’ 斜体

对字体的设置也可以用在title, xlabel, ylabel等中

图例

legend('a','Location','best') 图例位置放在最佳位置

更多的设置可以在绘图窗口中打开绘图工具，Inspector… 中查找

Various line types, plot symbols and colors may be obtained with

PLOT(X,Y,S) where S is a character string made from one element

from any or all the following 3 columns:

b blue . point - solid

g green o circle : dotted

r red x x-mark -. dashdot

c cyan + plus -- dashed

m magenta \* star (none) no line

y yellow s square

k black d diamond

w white v triangle (down)

^ triangle (up)

< triangle (left)

triangle (right)

p pentagram

h hexagram

在使用Matlab时，经常需要将得到的数值表达成二维或三维图像。

plot(vector1,vector2)可以用来画两个矢量的二维图，例如

x=1:0.1:2\*pi;

plot(x,sin(x))可以画正弦函数在0-2pi的上的图像。

plot函数可以接一些参数，来改变所画图像的属性（颜色，图像元素等）。下面是一些属性的说明

b blue（蓝色） . point（点） - solid（实线）

g green（绿色） o circle（圆圈） : dotted（点线)

r red（红色） x x-mark（叉号） -. dashdot (点画线）

c cyan（墨绿色） + plus(加号） -- dashed(虚线）

m magenta（紫红色） \* star（星号） (none) no line

y yellow（黄色） s square（正方形）

k black（黑色） d diamond（菱形）

v triangle (down)

^ triangle (up)

< triangle (left)

triangle (right)

p pentagram

h hexagram

例如，plot(x,y,'.r')表示用点来画图，点的颜色是红色。

plot函数可以接一些参数，来改变所画图像的属性（颜色，图像元素等）。下面是一些属性的说明

b blue（蓝色） . point（点） - solid（实线）

g green（绿色） o circle（圆圈） : dotted（点线)

r red（红色） x x-mark（叉号） -. dashdot (点画线）

c cyan（墨绿色） + plus(加号） -- dashed(虚线）

m magenta（紫红色） \* star（星号） (none) no line

y yellow（黄色） s square（正方形）

k black（黑色） d diamond（菱形）

v triangle (down)

^ triangle (up)

< triangle (left)

triangle (right)

p pentagram

h hexagram

Example

x = -pi:pi/10:pi;

y = tan(sin(x)) - sin(tan(x));

plot(x,y,'--rs','LineWidth',2,...

'MarkerEdgeColor','k',...

'MarkerFaceColor','g',...

'MarkerSize',10)

xlabel('x');

ylabel('y');

用Matlab画图时，有时候需要对各种图标进行标注，例如，用“+”代表A的运动情况，“\*”代表B的运动情况。

legend函数的基本用法是

LEGEND(string1,string2,string3, ...)

分别将字符串1、字符串2、字符串3……标注到图中，每个字符串对应的图标为画图时的图标。

例如:

plot(x,sin(x),'.b',x,cos(x),'+r')

legend('sin','cos')这样可以把"."标识为'sin'，把"+"标识为"cos"

还可以用LEGEND(...,'Location',LOC) 来指定图例标识框的位置

这些是Matlab help文件。后面一段是对应的翻译和说明

'North' inside plot box near top

'South' inside bottom

'East' inside right

'West' inside left

'NorthEast' inside top right (default)

'NorthWest inside top left

'SouthEast' inside bottom right

'SouthWest' inside bottom left

'NorthOutside' outside plot box near top

'SouthOutside' outside bottom

'EastOutside' outside right

'WestOutside' outside left

'NorthEastOutside' outside top right

'NorthWestOutside' outside top left

'SouthEastOutside' outside bottom right

'SouthWestOutside' outside bottom left

'Best' least conflict with data in plot

'BestOutside' least unused space outside plot

'North' 图例标识放在图顶端

'South' 图例标识放在图底端

'East' 图例标识放在图右方

'West' 图例标识放在图左方

'NorthEast' 图例标识放在图右上方（默认）

'NorthWest 图例标识放在图左上方

'SouthEast' 图例标识放在图右下角

'SouthWest' 图例标识放在图左下角

(以上几个都是将图例标识放在框图内）

'NorthOutside' 图例标识放在图框外侧上方

'SouthOutside' 图例标识放在图框外侧下方

'EastOutside' 图例标识放在图框外侧右方

'WestOutside' 图例标识放在图框外侧左方

'NorthEastOutside' 图例标识放在图框外侧右上方

'NorthWestOutside' 图例标识放在图框外侧左上方

'SouthEastOutside' 图例标识放在图框外侧右下方

'SouthWestOutside' 图例标识放在图框外侧左下方

（以上几个将图例标识放在框图外）

'Best' 图标标识放在图框内不与图冲突的最佳位置

'BestOutside' 图标标识放在图框外使用最小空间的最佳位置

还是用上面的例子

legend('sin','cos','location','northwest')可以将标识框放置在图的左上角。

Examples:

x = 0:.2:12;

plot(x,bessel(1,x),x,bessel(2,x),x,bessel(3,x));

legend('First','Second','Third');

legend('First','Second','Third','Location','NorthEastOutside')

b = bar(rand(10,5),'stacked'); colormap(summer); hold on

x = plot(1:10,5\*rand(10,1),'marker','square','markersize',12,...

'markeredgecolor','y','markerfacecolor',[.6 0 .6],...

'linestyle','-','color','r','linewidth',2); hold off

legend([b,x],'Carrots','Peas','Peppers','Green Beans',...

'Cucumbers','Eggplant')

xxxx=polyfit(text6(:,1),res(:,1),2)得到二次项的拟合系数

## 解微分方程:

function dy=F2(t,k)

Gm=3.986005e14;

vt=200;

mt=300-t;

ft=-vt/mt;

f=-Gm/((k(1)^2+k(2)^2+k(3)^2)^1.5);

dy=[k(4);k(5);k(6);f\*k(1)+ft;f\*k(2)+ft;f\*k(3)+ft];

先把高次的微分方程用y'=x的方式化成低次微分方程组

y0=[9092044.771852,1732113.220573,1732113.220573,-1566.513180,4453.807606,4453.807606];

[T0,Y0]=ode45('F2',[1:1:100],y0);

zbw0=Y0(1,:);

然后用初值再解即可.

format long: 多显示小数点后几位

## 字符串连接:

res=strcat( a,'set') %如果a是字符串

res=strcat(num2str(a),'set')%如果a是数字

cell型矩阵操作:k{1}

查看变量类型:class

[f,text]=xlsread('11.xls',1,'B1:G5807')

[res,ind]=sort(a,dim,mode)

dim 1 按列 dim 2 按行 mode 'scend' 'descend'是升序降序 ind: res=a[I];

转字符串 num2str(a)

save 's.mat' data1 data2(若要追加写-append)