Matlab Concise Notes

Gang Wang

Oct. 13, 2009

1. Introduction

Matlab stands for MATrix LABoratory and the objects of basic operations are vectors and matrices (user should be good at linear algebra). It is written in C. Matlab is an interactive/interpretative high-level language for scientific computing.

(1) interactive: line command, easy to debug
(2) interpretative: not pre-compiled, most time, must run within Matlab environment
(3) scientific computing: high level build-in algorithms for scientific computing

Algorithms + Data Structures = Programs

Algorithms: Let the best mathematician work for you.
Data structures: Everything is linear algebra (ODE, PDE…)
Programs: made it easy
Ax=B

Best tutorial: Matlab help files
(1) Use "help" to
>> help plot
(2) Search help files
(3) Some online tutorial: google search 'Matlab tutorial'
http://www.maths.dundee.ac.uk/~ftp/na-reports/MatlabNotes.pdf
http://www.ieee-uffc.org/ultrasonics/software/Matlab/
(4) Practice!

2. Matlab Demonstration

>> demos
(1) data visualization
Demos Matlab: More Examples
Graphics: 3d plot, volume visualization
3d drawing
Movie clips
(2) toolbox
demos Matlab toolbox
spline – construct a spline
image processing – edge detection (blood)
statistics – robust regression
(3) symbolic operation

```
x=sym('x')      % create a symbolic variable, x
f=1/[5+4*cos(x)]
ezplot(f)       % plot function f(x)
f1=diff(f)      % differentiate the function once
```

ezplot(f1)
f2=diff(f,2) % differentiate the function twice
ezplot(f2)
g=int(int(f2)) % integrate 2nd derivative twice
ezplot(g)

3. Matlab Environment
Path
Current working directory, know where your programs are
Workspace
script
echo or not echo (;)
Comments (%)

4. Basic Operations
(1) Basic Mathematics Operators
Example: Matlab as a calculator: 5*sin(2.5^(3-pi))+1/75

+ - * /
^ (power, eg. 2^3=8)
1.234e3 (=1.234*10^3)
abs(x)
sqrt(x)
log(x) -- natural log
log10(x) (base 10 log)
exp(x)
triangular functions: sin cos tan cot asin acos atan acot
(2) Some Constants
pi
inf
eps
NaN -- not a number is an important
Example
>>X=[1,2,5,NaN, 8, 9];
>>plot(X,'ro-');

(3) Logic operators
= = eq(,) ~= ne(,) < > =< >=
max(a,b)
min(a,b)
Example:
>> a=[1,2,3,4]
>> a = =2

5. Basic variables
(1) Real Numbers

```
>>pi
>>format long    format short        format long e          format short e
```

(2) Strings/Characters

```
string1=‘hello world’
string2=['hello ','world']
string3=num2str(3)    % convert from num to string
[string1,' 1 + 2= ',string3]
A=[1,2,4,8]
disp(A)
disp(string1)
sprintf('this is %d + %d = %d',1,2,1+2)      %Write formatted data to string.
sprintf('this is %d + %d = %10.4f',1,2,1+2)
sprintf('this is %d + %d \n= %10.4f',1,2,1+2)
disp(sprintf('this is %d + %d = %10.4f',1,2,1+2))
```

(3) load/save variables

```
>> clear all
>> myMatrix=[1,3, 5, 7, 9]
>> save('savedmyMatrix.mat','myMatrix')      % save as .mat file
>> load('savedmyMatrix.mat')                 % load variables stored in .mat file
>> load savedmyMatrix    % the same
>> who                                       % check variable in workspace
```

6. Basic Data Structurs

(1) Vectors and matrices

```
x=[ 1 2 3 4 5 6 7 8 9 10]
x=1:10
x=1:2:10
x=linspace(1,10,3)
A=[1 2 3 ; 4 5 6 ; 7 8 9 ]
A(3,1)
```

Matrix operation

```
+    -    *    /
A*B
(m,n) (n*p)    (m*p)
```

Example: Solve AX=B

```
A=[1,2;3,4]
B=[5;6]
X=inv(A)*B
X=A\B    % partial pivoting
```

```
size(A)
inv(A)
```

A*A’
trace(A)
det(A)
eye(3)
zeros(2,3)
[i,j]=find(A==2)
eig(A)

see matlab demos: matrices, basic operation
(2) string is also stored in matrix structure

(3) Structural Array
1. Construct the structural array
>>GMXEmployee(1).ID=1234
>>GMXEmployee(1).Name='Gang Wang'
>>GMXEmployee(2).ID=567
>>GMXEmployee(2).Name='Miao Zhang'
>>GMXEmployee.ID(1)
2. Make the query
>>GMXEmployee(1).Name
ans =

Demos: matlab-> language-> structures

(4) Cells
Enclose the cell subscripts in parentheses using standard array notation. Enclose the cell contents on the right side of the assignment statement in curly braces, "{}." For example, create a 2-by-2 cell array A.

>>A(1,1) = {[1 4 3; 0 5 8; 7 2 9]};
>>A(1,2) = {'Anne Smith'};
>>A(2,1) = {3+7i};
>>A(2,2) = {-pi:pi/10:pi};

Try to calculate A(2,1)+1, oops, function '+' not defined for variables of class 'cell'. What should we do?
Convert cell to matrix first, then do calculation
>>**cell2mat**(A(2,1))+1
ans =

    4.0000 + 7.0000i
>> **cell2mat**(A(1,2))
Advantage of using cells:
1. assemble different data types
2. assemble data of different length, especially string variables

B(1)={'this is a long sentence'}

B(2)={'this is short'}

Colon notation

Generate a list with start:step:end

```
a=1:2:10        %(step 2, from 1 to 10)
a=1:10          %(step 1 as default)
```

Retrieve sub-block of a matrix

```
A=[1,2,3; 4,5,6; 7,8,9]
A(:,1)        %the first column of matrix A
A(1,:)        %the first row of matrix A
A(2:3, [1,3])=3
```

SUM operation

```
>> A = [1:3; 4:6; 7:9]
A =
1    2    3
4    5    6
7    8    9
>> s = sum(A)
  s = 12 15 18
>> ss = sum(sum(A))
  ss = 45
```

Matrix entry-to-entry operation

```
A=[1 2 3];
B=[3 4 5];
```

1. Dot product: A.*B

```
ans =
        3         8        15
```

Compare it with A'*B = $\begin{pmatrix}1\\2\\3\end{pmatrix}\begin{pmatrix}3 & 4 & 5\end{pmatrix}$

```
ans =
        3         4         5
        6         8        10
        9        12        15
```

and A*B'= $\begin{pmatrix}3 & 4 & 5\end{pmatrix}\begin{pmatrix}1\\2\\3\end{pmatrix}$ =26

2. Dot division: A./B

```
ans =
       0.3333       0.5000       0.6000
```

3. Dot power : A.^2
ans =
1 4 9

7. Building Blocks of Structured Programming
(1) IF statement

```
if expression
   statements

else
   statements
end
```

The statements are executed if the real part of the expression has all non-zero elements. The ELSE and ELSEIF parts are optional. Zero or more ELSEIF parts can be used as well as nested IF's. The expression is usually of the form expr rop expr where rop is ==, <, >, <=, >=, or ~=.

(2) **FOR Loops** Repeat statements a specific number of times.
The general form of a FOR statement is:

```
for variable = expression
    statement
    statement
end
```

example:

```
for i=['a','b', 'c']
  disp(i)
end
```

The BREAK statement can be used to terminate the loop prematurely.
Example:

```
for i=1:10
    disp(i)
    if i ==5
        break
    end
end
```

(3) **WHILE Loops:** Repeat statements an indefinite number of times.
The general form of a WHILE statement is:

```
while expression
   statements
end
```

The statements are executed while the real part of the expression has all non-zero elements. The expression is usually the result of expr rop expr where rop is ==, <, >, <=, >=, or ~=.

Example:
```
eps = 1;
while (1+eps) > 1
      eps = eps/2
end
eps = eps*2
```

(4) SWITCH CASE: Switch among several cases based on expression.

Example:
```
METHOD='LINEAR';
switch lower(METHOD)
      case {'linear','bilinear'}
         disp('Method is linear')
      case 'cubic'
         disp('Method is cubic')
      case 'nearest'
           disp('Method is nearest')
      otherwise
           disp('Unknown method.')
end
```

8. Matlab Scripts

A script is an m-file without the function declaration at the top. Be aware that all the variables in a scripts can be altered.

hit : (1) always use “clear all” to start a new script

(2) make sure the path saved

9. Function

Example:
```
function [A] = myarea(a,b,c)
% Compute the area of a triangle whose
% sides have length a, b and c.
% Inputs:
% a,b,c: Lengths of sides
% Output:
% A: area of triangle
% Usage:
% Area = area(2,3,4);
```

```
% Written by dfg, Oct 14, 1996.
s = (a+b+c)/2;
A = sqrt(s*(s-a)*(s-b)*(s-c));


>> help area
>> Area = area(10,15,20)
```

Change function declaration to
function [A,s] = area(a,b,c)
then, the returned variables are A and s

```
>> Area = area(10,15,20)              % by default, the 1st returned variable is assigned
>> [Area, hlen] = area(10,15,20)
```

10. File Operation
READ/WRITE a file

(1) fopen, fclose

```
>> fid1 = fopen('sound.dat','r');
>> fclose(fid1)
```

(2) fgetl: Read line from file, discard newline character

Example

```
fid=fopen('fgetl.m');
while 1
      tline = fgetl(fid);
      if ~ischar(tline), break, end
      disp(tline)
end
fclose(fid);
```

(3) fprintf: Write formatted data to file

Example:

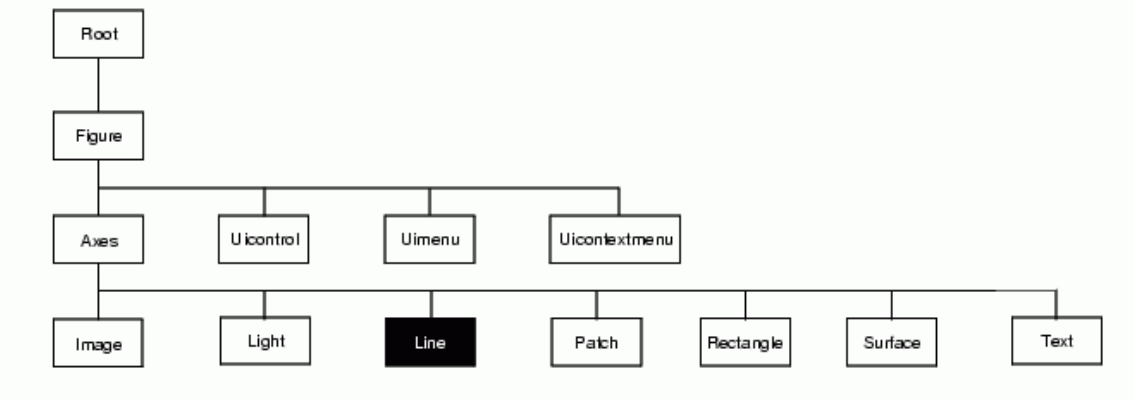
```
x = 0:.1:1;
y = [x; exp(x)];
fid = fopen('exp.txt','w');
fprintf(fid,'%6.2f %12.8f\n',y);
fclose(fid)
```

(4) fwrite: Write binary data to a file
(5) fread: Read binary data from file

Matlab Graphics.

Matlab is a powerful tool to visualize data.
1. Two Dimensional (2-D) Plot Y=f(X)
2. Three Dimensional (3-D) Plot Z=f(X, Y)
3. Volume Data (4-D) Visualization X, Y, Z, V(X, Y, Z)

Object Hierarchy



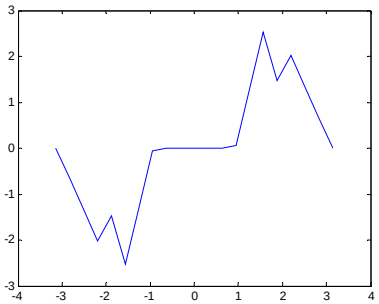
11. Two Dimensional (2-D) Plot

(1) Linear 2-D **plot** command Syntax

```
Syntax
plot(Y)
plot(X1,Y1,...)
plot(X1,Y1,LineSpec,...)
plot(...,'PropertyName',PropertyValue,...)
h = plot(...)
```

Example:

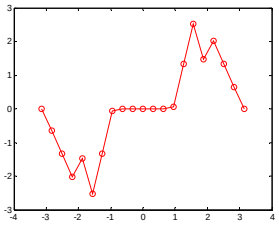
```
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
plot(x,y)
```

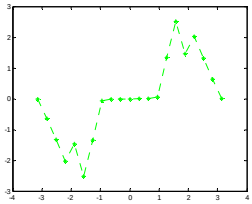


Use File-> Save; Edit->Copy Figure in the control bar to save/copy the figure
Use arrow and double click the graphic to open Graphic Property panel.

(2) Use short symbols to specify COLOR, MARKER, LINE TYPE

```
plot(x,y,'ro-');      % r – red; o—circle, - solid line
plot(x,y,'g*:');    % g – green; * - star, : dot line
```





Color Specifiers

Specifier	Color
r	red
g	green
b	blue
c	cyan
m	magenta
y	yellow
k	black
w	white

Marker Specifiers

Specifier	Marker Type
+	plus sign
o	circle
*	asterisk
.	point
x	cross
s	square
d	diamond
^	upward pointing triangle
v	downward pointing triangle
>	right pointing triangle
<	left pointing triangle
p	five-pointed star (pentagram)
h	six-pointed star (hexagram)

Line Style Specifiers

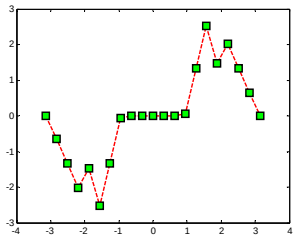
Specifier	Line Style
-	solid line (default)
--	dashed line
:	dotted line
-.	dash-dot line

(3) Specification of other graphic properties

You can also specify other line characteristics using graphics properties.
The most commonly used graphic properties are:
"Color" -- Three ways to assign color in Matlab
"LineWidth" - specifies the width (in points) of the line.
"MarkerEdgeColor" - specifies the color of the marker or the edge color for filled markers (circle, square, diamond, pentagram, hexagram, and the four triangles).
"MarkerFaceColor" - specifies the color of the face of filled markers.
"MarkerSize" - specifies the size of the marker in units of points.

Examples:

```
h=plot(x,y,'--rs','LineWidth',2, 'MarkerEdgeColor','k', 'MarkerFaceColor','g','MarkerSize',10)
```



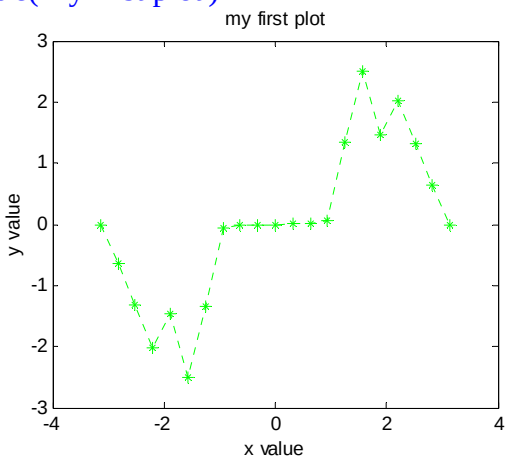
Important Concept: Use Object Handle to Operate
gca -- get current axes handle
gcf -- get current figure handle

Examples:

```
set(gca,'Color','k')      % gca – get current axes
set(gcf,'Color','y')      % gcf – get current figure
set(gcf,'Color','yellow')
set(gcf,'Color',[1,0,0])
set(gcf,'Color',[0,0.5,0.5])    % another fancy color

h=plot(x,y,'g*:');    % set h as handle of the line object
set(h, 'Color', 'b');    % set the handle object (line) to blue
get(h,'Color'); % use get command to get color property of the handle object (i.e. line)
get(h,'LineWidth')
set(h,'MarkerSize',5)
```

(4) More Operations on the Figure

```
h1=plot(x,y,'g*:');
xlabel('x value')
ylabel('y value')
title('my first plot')
```




```
grid on
grid off
h2=plot(x,-y,'bo-');    % first figure was erased!! What should we do?
```

HOLD ON to plot multiple lines in the same figure, until you HOLD OFF

```
h1=plot(x,y,'g*:');
hold on;
h2=plot(x,-y,'bo-');
```

Now put in label, and change the font size of the lable

```
hh=ylabel('y value')
set(hh,'FontSize',14)
```

Put a text on the figure

```
text(0,5,'my comments')
```

Put on legend

```
legend('1st Line','2nd Line',3) % note that the last entry is the position; try 1, 2, 3, 4
```

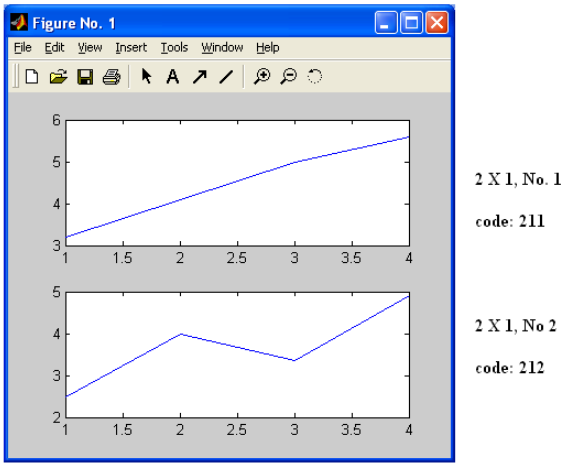
Save current figure to windows bitmap file

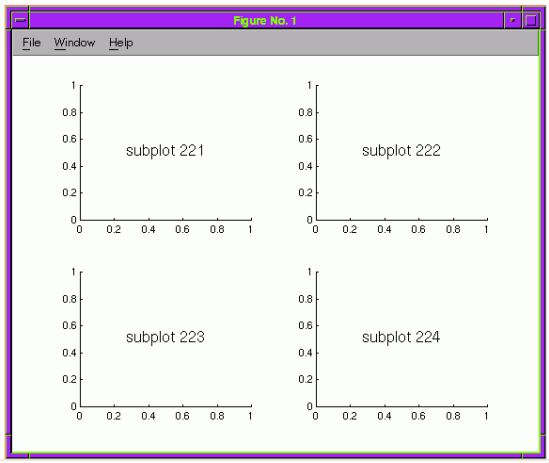
```
saveas(gcf, 'output', 'bmp')
```

You can save to other formats: tiff, jpeg, eps, pdf ….

(5) subplot

```
clf                    % clear figure
income = [3.2 4.1 5.0 5.6];
outgo = [2.5 4.0 3.35 4.9];
subplot(2,1,1); plot(income)
subplot(2,1,2); plot(outgo)
```





You can also make 2x2 subplots like this

(6) Other 2D plot commands:

```
loglog(    )
semilogx(    )
semilogy(    )
```

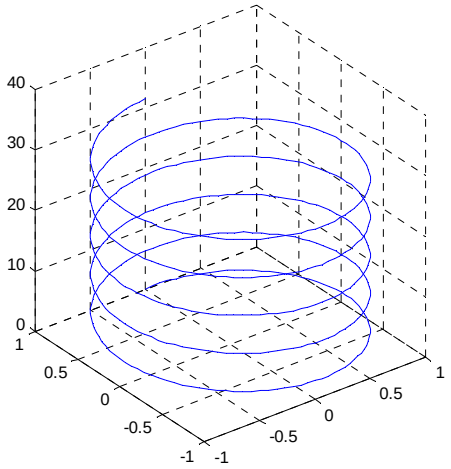
12. Three Dimensional (3D) Plot

(1) Linear 3-D plot command: plot3

Syntax
plot3(X1,Y1,Z1,...)
plot3(X1,Y1,Z1,LineSpec,...)
plot3(...,'PropertyName',PropertyValue,...)
h = plot3(...)

Example:

```
t = 0:pi/50:10*pi;
h=plot3(sin(t),cos(t),t)
grid on
axis square
```




```
set(h,'Color','r','LineWidth',3,'LineStyle',':')
```

change view angle: [1,1,1] by default, you can manually rotate the view

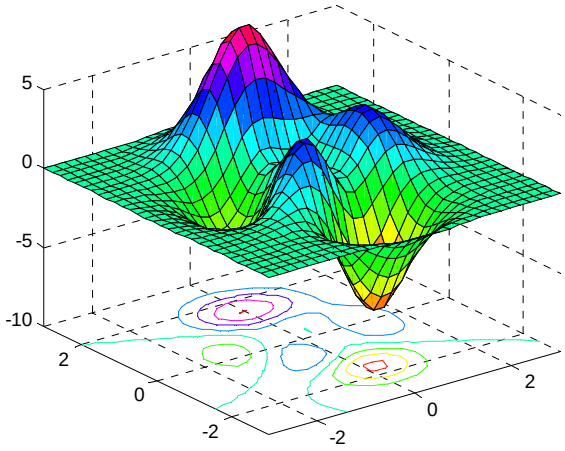
```
view([1,0,1])
```

(2) 2D surface plot

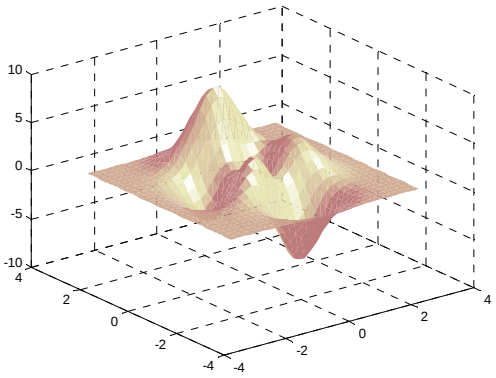
SURF : 3-D shaded surface plot
SURFC: 3-D shaded surface plot with contour
SURFL: 3-D shaded surface with lighting.

Display a surface and contour plot of the peaks surface.

```
[X,Y,Z] = peaks(30);    % peaks is just a 3D function
surf(X,Y,Z)                    % can use surf(X,Y,Z), then the plot is without contour
colormap hsv                   % choose colormap scheme
axis([-3 3 -3 3 -10 5]) % specify the range of axis
```

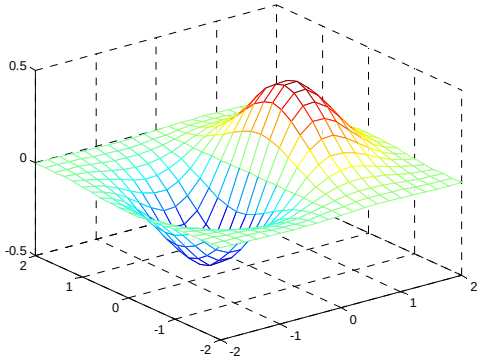


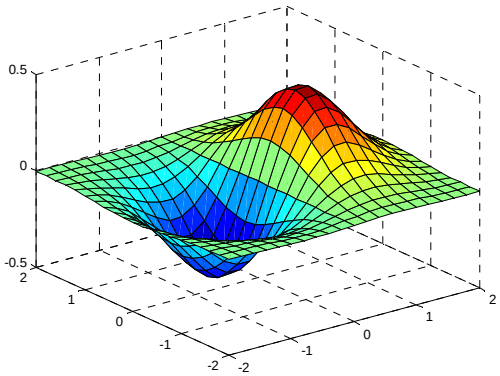
Try
surf(X,Y,Z)
Add on color bar
colorbar % show colorbar
colormap jet % change colormap scheme to jet
Change shading to interpolated scheme
shading interp;
colormap(pink);



Other important commands: meshgrid; mesh;

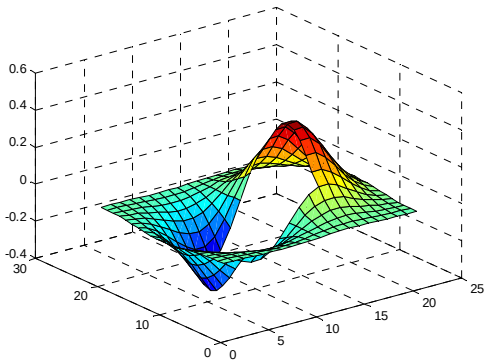
```
[X,Y] = meshgrid(-2:.5:2, -2:.5:2);    % construct X,Y grids
Z = X .* exp(-X.^2 - Y.^2);
mesh(X,Y,Z)
surf(X,Y,Z)
```

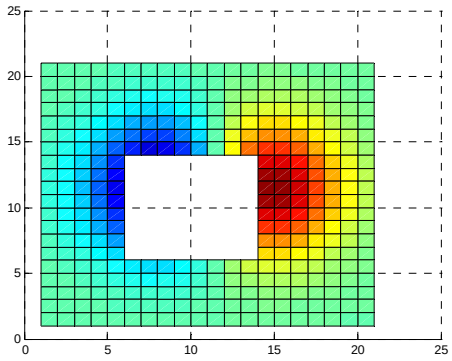




Note: NaN data is NOT visualized!1

```
% assign some z values as NaN in the middle rectangular area
Z(7:13, 7:13)=NaN;
surf(Z)
```





This example shows a hole in the figure!

Demostration

>> demos

Choose Matlab -> Graphics

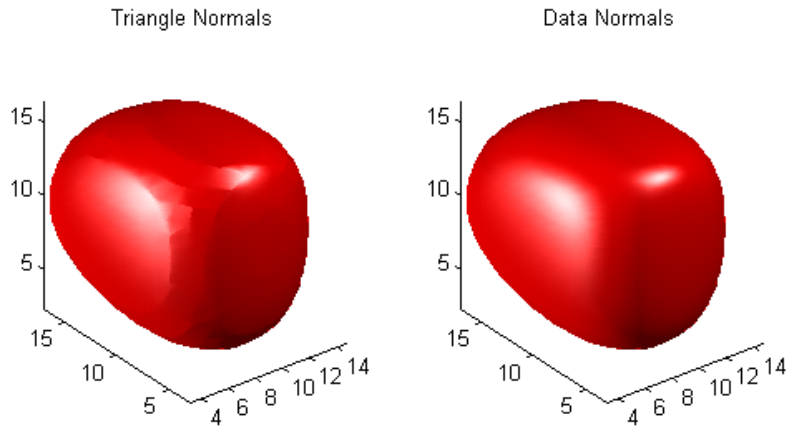
3D plot

Transparency

13. Volume Data (4-D) Visualization

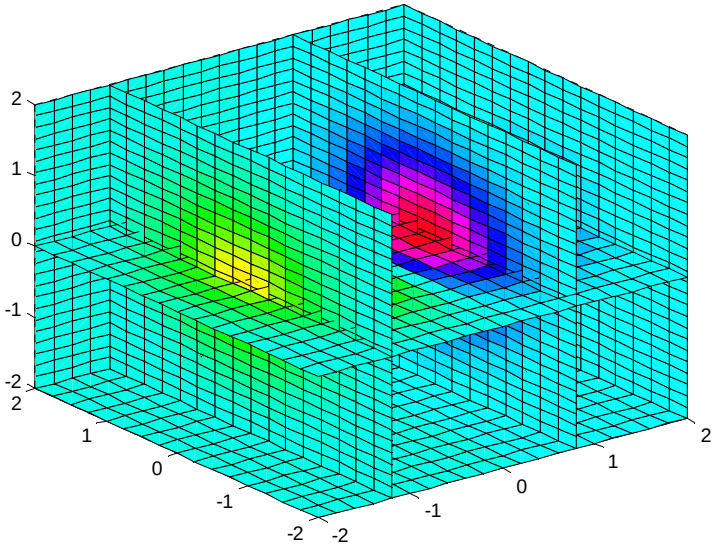
Example 1: isosurface, isonormal

```
data = cat(3, [0 .2 0; 0 .3 0; 0 0 0], [.1 .2 0; 0 1 0; .2 .7 0], [0 .4 .2; .2 .4 0;.1 .1 0]);
data = interp3(data,3,'cubic');

subplot(1,2,1)
p1 = patch(isosurface(data,.5), 'FaceColor','red','EdgeColor','none');
view(3); daspect([1,1,1]); axis tight
camlight; camlight(-80,-10); lighting phong;
title('Triangle Normals')

subplot(1,2,2)
p2 = patch(isosurface(data,.5), 'FaceColor','red','EdgeColor','none');
isonormals(data,p2)
view(3); daspect([1 1 1]); axis tight
camlight;    camlight(-80,-10); lighting phong;
title('Data Normals')
```



Example 2 Slices of volumetric data

```
[x,y,z] = meshgrid(-2:.2:2,-2:.25:2,-2:.16:2);
v = x.*exp(-x.^2-y.^2-z.^2);
xslice = [-1.2,.8,2]; yslice = 2; zslice = [-2,0];
slice(x,y,z,v,xslice,yslice,zslice)
colormap hsv
```



Example 3 Isosurface, isocaps, coneplot, and streamlines of wind data

```
load wind                    % load wind data, it contains u  v  w  x  y  z  variables
spd = sqrt(u.*u + v.*v + w.*w);
p = patch(isosurface(x,y,z,spd, 40));
isonormals(x,y,z,spd, p)
set(p, 'FaceColor', 'red', 'EdgeColor', 'none');
p2 = patch(isocaps(x,y,z,spd, 40));
```