CNBC/IGERT MATLAB Minicourse: Lecture 3

David S. Touretzky

January 2007

Map the M: Drive

In the toolbar at the bottom of the screen, click on the AFS Client icon (it looks like a yellow padlock.)

Select “Drive Letters”

Click “Add”

Choose drive M:

Use this path:
\afs\andrew.cmu.edu\usr\dst\matlab

Multidimensional Arrays

MATLAB supports arrays with more than two dimensions.

```
m = rand(4,3,2)
```

```
whos m
size(m)
```

Matrix multiplication does not apply to higher dimensional arrays.

Array concatenation can’t be done with [ ] notation for more than 2 dimensions, so use **cat**(dim,A,B) instead.

```
cat(1,m,m)
cat(2,m,m)
cat(3,m,m)
```

Sparse Matrices

Sparse matrices provide an efficent means to store matrices that are mostly empty.

```
s = sparse(1000,1000)
s(2,11) = 2
s(992,875) = 3
s(875,992) = 4.7

whos s
```

All arithmetic operations work on sparse as well as full matrices.

```
s * 10

s’ * s

s * s’
```

Why is s+10 *a bad idea?*

Cell Arrays

Ordinary arrays contain elements of a single type, such as doubles or chars.

Cell arrays contain objects of heterogeneous types.

```
b = {1 ’two’ [3 3 3]}
```

Cell arrays offer greater flexibility.

Drawback: cell arrays require much more storage, because each element requires a separate pointer and type information.

```
ra = [1 2 3 4];
ca = {1 2 3 4};
whos ra ca

ra(2:3)
ca{2:3}
```

Structure Arrays

MATLAB supports “structure arrays” with named fields.

```
a.name = ’John Smith’
a.age = 35
a.department = ’Accounting’
whos a
```

The array a is a scalar (1x1) structure array.

```
f = fieldnames(a)
f{1}
```

The **what** and **get** functions return structures.

```
w = what
w.m

p = get(gcf)
```

Structure Arrays (cont.)

Structure arrays can contain more than one element.

All elements have the same set of field names; some fields may be empty.

```
a(2).name = ’Mary Brown’
a(2).seniority = 8


whos a

a(1)

a(2)
```

Call-by-Value Semantics

MATLAB uses call-by-value semantics, making it impossible for functions to modify their arguments.

In C, integers and reals are passed by value, but arrays are passed by reference, making them modifiable.

The following doesn’t work:

```
function birthday(emp)
    emp.age = emp.age + 1
```

```
birthday(a(1))

a(1)
```

Return Modified Structures

Modified arrays or structures must be returned, and assigned back to the original variable.

Otherwise the modifications are lost.

```
function emp = birthday(emp)
  emp.age = emp.age + 1;
```

```
a(1)

a(1) = birthday(a(1));

a(1)
```

Efficiency Considerations

When an array or structure is passed as an argument, MATLAB doesn’t necessarily copy it.

Objects are passed to functions by reference, but are copied if the function modifies the argument.

The modify-and-return approach is not an efficient way to maintain large objects, due to excessive copying.

Alternative solution: store values in a global variable and let functions modify that variable directly.

Property Arguments

The **plot** function and many other graphics functions accept property/value pairs as extra arguments.

```
clf, hold on

plot(rand(5,1), ’Color’, [0 0.5 0.8])

plot(rand(5,1),’r’,’LineWidth’,8, ...
        ’LineStyle’,’--’)

plot(rand(5,1),’Color’, [0.7 0.9 0.2], ...
        ’LineWidth’, 6, ’Marker’,’^’)
```

Note: the ... notation is used to continue a Matlab statement across multiple lines.

GUI Facility

UIcontrol objects include pusbuttons, sliders, pop-up menus, and radio buttons.

```
clf

hb = uicontrol(’Style’,’pushbutton’)

set(hb, ’String’, ’Foo’)

set(hb, ’BackgroundColor’, [0.2  0.6  1])

c = 0;

set(hb, ’Callback’, ’c=c+1, datestr(now)’)

set(hb, ’Style’, ’checkbox’)
```

Units Property

The Units property controls whether certain subsequent properties are interpreted as pixels, points, or percentage of screen size (“normalized” units).

```
set(hb,’Units’,’pixels’, ...
     ’Position’,[100 100 80 25])

set(hb,’Units’,’normalized’, ...
     ’Position’,[0.5 0.5  0.25 0.25])
```

Pop-up Menus and List Boxes

This is long; type it into a file:

```
hp = uicontrol(’Style’,’popup’, ...
  ’String’,{’eeny’,’meeny’,’miny’,’moe’}, ...
  ’Units’,’normalized’, ...
  ’Position’,[0.2 0.2 0.3 0.1], ...
  ’BackgroundColor’, [0.8 0.8 0.5], ...
  ’ForegroundColor’,[0.5 0.5 0.2])

set(hp,’Callback’,’get(gcbo,’’Value’’)’)
```

The **gcbo** function returns the object whose callback function is currently executing.

```
set(hp,’Position’,[0.2  0.2  0.3  0.2])

set(hp,’Style’,’listbox’)
```

Sliders

```
hs = uicontrol('Style','slider', ...
        'Position', [200 200 150 20], ...
        'Callback','get(gcbo,''value'')')
```

```
set(hs,'CallBack', ...
  'set(gcf,''Color'', ...
     [0  0  get(gcbo,''Value'')])')
```

In practice, the Callback string is usually a call to some user-written function with **gcbo** as the argument.

All the work is done inside the function.

GUIDE

GUIDE is the GUI Design Environment.

Allows interactive layout of a GUI window, including menus, graphics, text boxes, etc., using “drag and drop” techniques.

```
doc guide

guide
```

Creates a .fig file to store layout information.

Creates an editable .m file to load the .fig file and hold associated callback routines.

Image Data

```
clear all

load durer

whos

image(X)

colormap(map)

axis image

axis off

set(gca, ’Position’, [0 0 1 1])

colormap(‘hot’)

brighten(0.7)
```

Reading Image Files

```
imfinfo('M:\brain.jpg')

brain = imread('M:\brain.jpg');
whos brain
```

The uint8 datatype holds unsigned bytes.

```
image(brain)

colormap(bone)

axis image

colormap(bone(256))

zoom on
```

Mouse Input

```
getline('closed')
```

Click the left button to enter points. Click the right mouse button to end. Return value is a matrix of points defining the polygon.

```
p1 = getptr(gcf)

setptr(gcf,'hand')

p2 = getptr(gcf)

set(gcf,p1{:})
```

The p{:} notation expands the contents of the cell array p into multiple arguments to set.

Image Manipulation

```
function bmap(im)
  clf, zoom on

  colormap([bone(128); autumn(128)])
  d = double(im)/2;
  image(d),   axis image, axis off

  coords = getline(’closed’);
  [x,y] = meshgrid(1:size(im,1), ...
                1:size(im,2));
  z = inpolygon(x,y, ...
            coords(:,1), coords(:,2));
  d(z) = d(z) + 128;

  image(d), axis image, axis off
```

Call it like this: bmap(brain)

Toolboxes

Toolboxes contain collections of related functions that extend the basic MATLAB language.

The **matlab** toolbox contains a variety of libraries that implement MATLAB features such as graphing and matrix functions.

```
doc graph2d
doc stats
```

The **images** and **stats** toolboxes contain routines for image processing and statistical calculation.

The **ver** command displays version information for all the toolboxes currently installed.

```
ver
```

Search Path

You can control the search path MATLAB uses to find functions and data files.

```
path

addpath('M:\bump')

what bump

bump
```

Thinking in MATLAB

This is file M:\think.m

```
pts = 0:pi/160:12*pi;

data = sin(pts) + sin(0.3*pts) + cos(3*pts);
clf, hold on, plot(data)

thresh = mean(data) + std(data);
b = data > thresh;

bstart = ~b(1:end-1) & b(2:end);
bend = b(1:end-1) & ~b(2:end);

pstart = find(bstart);
pend = 1+find(bend);

xcoords = [pstart; pend; NaN*pstart];
ycoords = repmat(thresh,size(xcoords));

plot(xcoords(:),ycoords(:),’r’,’Linewidth’,3)
```

Learn More About MATLAB

- Browse the on-line documentation.
- “demos” gives you a large collection of demos / mini-tutorials.
- See the mathworks.com web site for latest Matlab news.
- Amazon.com lists a whopping 633 results for books with “MATLAB” in the title.
- Usenet newsgroup:
  `comp.soft-sys.matlab`
- Read some code! Much of MATLAB is written in MATLAB.