

## CNBC Matlab Mini-Course

David S. Touretzky  
October 2021

Day 3: The Really Cool Stuff

1

## Sparse Matrices

Sparse matrices provide an efficient 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
```

4

## Multidimensional Arrays

Matlab supports arrays with more than 2 dimensions.

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

```
whos m
```

```
size(m)
```

Matrix multiplication does not apply to higher dimensional arrays.

2

## Math on Sparse Matrices

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

```
s * 10
```

```
s' * s
```

```
s * s'
```

*Why is  $s+10$  a bad idea?*

5

## Array Concatenation

Array concatenation can't be done with `[]` for higher dimensional arrays, so use `cat(dim, A, B)` instead.

```
d1 = cat(1, m, m)
```

```
d2 = cat(2, m, m)
```

```
d3 = cat(3, m, m)
```

3

## Cell Arrays

A "cell" is a container that holds both data and type information. Arrays of cells ("cell arrays") can contain objects of heterogeneous types:

```
b = 5
```

```
c = num2cell(5)
```

```
whos
```

```
b(3) = 3.5
```

```
c(3) = 3.5 ← Error: 3.5 is not a cell
```

```
c(3) = num2cell(3.5)
```

```
whos
```

```
c(4) = cellstr('rutabaga')
```

6

## { } Constructor Makes Cell Arrays

```
d = { 10 20 ; 30 40 }
```

```
c(2,1) = {3}
```

```
c(2, 2:4) = { 'parsnip' -5.1 2+sqrt(-9) }
```

Transpose works: 

```
{ 1 rand(5,3) sqrt(-2) }'
```

Arithmetic doesn't:

```
{ 1 2 3 } + { 4 5 6 }
```

7

## Cell Arrays Require More Storage

```
ra = [ 1 2 3 4 ]
```

```
ca = { 1 2 'three' 4 }
```

```
whos ra ca
```

*Each cell in the array requires separate type and and shape information.*

10

## Displaying and Concatenating Cells

Individual cells are always displayed enclosed either in brackets [ ] or quotes ' '.

```
foo = { 3.4 'green' -9 }
```

```
bar = { 2 rand(5) 3 }
```

```
[foo bar] Concatenating arrays of cells
```

```
[foo ; bar]
```

```
{ 'a' 'bcd' 'ef' } An array of cells
```

```
[ 'a' 'bcd' 'ef' ] Strings are arrays of chars
```

8

## { } Indexing Gives Cell Contents

```
a = { 10 20 'thirty' 40i }
```

```
a(2) parens
```

```
a{2} braces
```

Slicing returns an array:

```
a(2:3)
```

Slicing with { } returns multiple values:

```
a{2:3}
```

11

## { } And Assignment

When { } indexing is used on the left hand side of an assignment, the content of the cell is modified.

```
foo(4) = { 7 }
```

```
foo{5} = 11
```

9

## Extracting Cell Contents

```
ca = { 1 2 3 }
```

```
5 + ca Error: can't do arithmetic on cells.
```

```
5 + ca{ : } Error: too many arguments.
```

```
plus(5, 1, 2, 3)
```

```
5 + [ ca{ : } ]
```

```
lookfor cell
```

```
5 + cell2mat(ca)
```

```
[cx, cy] = ca{2 : 3}
```

12

## Property Arguments

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

```
props = {'Color', [0 0.5 0.8], 'LineWidth', 8, ...  
        'LineStyle', '-.' }
```

```
plot(rand(5,3), props{ : } )
```

13

## Multi-Element Structure Arrays

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)  
a.age  
{ a.age }
```

16

## Structure Arrays

Matlab provides "structure arrays" with named fields.

```
clear a  
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}
```

14

## Call-by-Value Semantics

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

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

17

## Returning Structure Arrays

The **what** and **get** functions return structure arrays.

```
w = what           Value is a structure array  
mfiles = w.m      Value is a cell array  
whos  
  
p = get(gcf)
```

15

## Call-by-Value (cont.)

The following doesn't work. Put this in `birthday.m`:

```
function birthday(employee)  
    employee.age = employee.age + 1  
end
```

Try it and see. Type the following in the console:

```
birthday(a(1))  
a(1)
```

18

## Returning Modified Structures

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

Otherwise the modifications are lost.

```
function employee = birthday(employee)
    employee.age = employee.age + 1
end
```

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

19

## GUI Facility (cont.)

```
c = 0;
set(hb, 'Callback', 'c=c+1, datestr(now)')
```

*click the button several times*

```
set(hb, 'Style', 'checkbox')
```

22

## 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 the functions modify that variable directly instead of passing values as arguments.

20

## Units Property

The Units property controls whether certain subsequent properties are interpreted as pixels, points, or percentage of screen or window 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])
```

23

## GUI Facility

UIControl objects include pushbuttons, sliders, pop-up menus, and radio buttons.

```
clf
hb = uicontrol('Style', 'pushbutton')
set(hb, 'String', 'Foo')
set(hb, 'BackgroundColor', [0.2 0.6 1])
```

21

## Pop-Up Menus and List Boxes

*Put this in a script:*

```
clf
hp = uicontrol('Style', 'popup', ...
    'String', {'eeny', 'meeny', 'miney', 'moe'}, ...
    'Units', 'normalized', ...
    'Position', [0.2 0.2 0.3 0.1], ...
    'BackgroundColor', [0.8 0.8 0.5], ...
    'ForegroundColor', [0.1 0 0.95])
set(hp, 'Callback', 'get(gcbo, 'Value')')
```

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

24

## Sliders

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

After trying the above, try this:

```
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.

25

## Image Data

```
clf reset, 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)
```

28

## HHsim's GUI Interface

```
cd /afs/andrew/usr/dst/matlab/hhsim  
hhsim
```

*Click on the stim1 button.*

```
cd ~
```

26

## Reading Image Files

```
cd ~  
!wget www.cs.cmu.edu/~dst/Tutorials/Matlab/brain.jpg  
brain = imread('brain.jpg') ;
```

```
whos brain
```

The uint8 datatype holds unsigned bytes.

```
image(brain)  
colormap(bone)  
axis image  
colormap(bone(256))  
zoom on
```

29

## GUIDE

GUIDE is the GUI Design Environment

Tool that 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.

27

## Mouse Input

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

Click the left button to enter points. Click the right 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 p1{:} notation expands the contents of the cell array p1 into multiple arguments to set.

30

## Image Manipulation

```
function bmap(im)
    clf
    colormap([bone(256); autumn(256)])
    d = double(im) ;
    image(d), axis image, axis off
    coords = getline('closed');
    [x,y] = meshgrid(1:size(im,2), 1:size(im,1));
    z = inpolygon(x, y, coords(:,1), coords(:,2));
    d(z) = d(z) + 256;
    image(d), axis image, axis off
end
```

Call it like this:  
bmap(brain)  
or bmap(X)

31

## Search Path

You can control the search path Matlab uses to search for functions and data files.

```
path
!mkdir mystuff
!mv bmap.m mystuff
bmap(brain) Matlab can't find it anymore
addpath('mystuff')
what mystuff
bmap(brain)
```

34

## 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 calculations.

32

## Where Do You Go From Here?

- Pick a dataset of interest and explore various ways to plot it.
- Try some data analysis examples in next week's class.
- Try out some of the built-in demos.
- Spend some time browsing the documentation to learn more about the statistics toolbox or the handle graphics system.
- Purchase a Matlab book and work through the examples.

35

## Version Info

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

This is especially useful when reporting a bug in Matlab or checking whether the version you are running is the latest one.

```
ver
```

```
a = ver('stats')
```

33