Computing with MATLAB™

Part 3

Icahn School of Medicine at Mount Sinai

SBCNY

Outline

Using MATLAB for data manipulation and analysis

Useful MATLAB built-in functions

e.g. mean, maximum, standard deviation
determining array locations that fit particular criteria

Strategies for using `if` statements

Exporting and importing results

Useful built-in MATLAB functions

```
>> a = [1,5,2,4,3]
a =
     1     5     2     4     3
>> b = mean(a)
b =
     3
```

[calculates the average]

```
>> c = std(a)
c =
    1.5811
```

[calculates standard deviation]

```
>> d = max(a)
d =
     5
>> e = sum(a)
e =
    15
>> f = sqrt(a)
f =
    1.0000    2.2361    1.4142    2.0000    1.7321
```

[sqrt = square root
calculate element by element]

```
>> g = exp(a)
g =
    2.7183  148.4132    7.3891   54.5982   20.0855
```

[exp(x) = e^x]

```
>> h = max(sqrt(exp(a)))
h =
   12.1825
```

[functions can easily be combined]

3

Useful built-in MATLAB functions

```
a =
     1     5     2     4     3
>> [d,index] = max(a)
d =
     5
index =
     2
```

[d holds the maximum value
index holds its location in the array]

Next we'll see how this can be used

```
>> j = find(a > 2)
j =
     2     4     5
```

[returns indices of all elements > 2]

```
>> A = [1,5,2;3,0,1;2,4,7]
A =
     1     5     2
     3     0     1
     2     4     7
>> k = max(A)
k =
     3     5     7
```

[maximum of each column is computed]

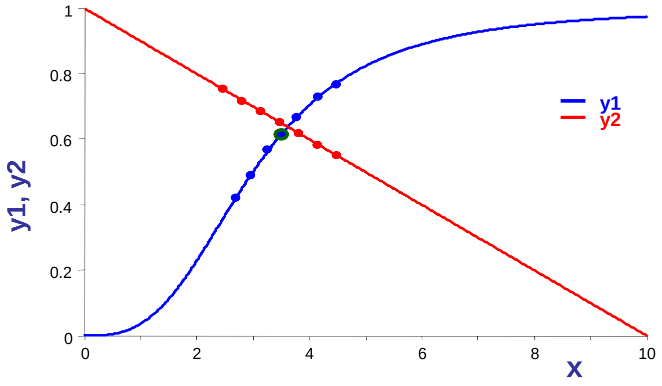
```
>> l = max(max(A))
l =
     7
```

[this returns the overall maximum]

4

A practical example

Where do two curves cross?



Where the curves cross, y1=y2, so y1-y2 = 0
But what if the values are never exactly equal?

```
>> [dummymin,index] = min(abs(y1-y2)) ;
>> crossingpoint = x(index)
>> plot(x(index),y1(index),’go’,’LineWidth’,3)
```

Built in matlab functions

Combining functions with concatenation

```
>> a = (1:10)' ;
>> A = [a,sqrt(a),a.^2,sin(a)]
A =
  1.0    1.0000    1.0   0.8415
  2.0    1.4142    4.0   0.9093
  3.0    1.7321    9.0   0.1411
  4.0    2.0000   16.0  -0.7568
  5.0    2.2361   25.0  -0.9589
  6.0    2.4495   36.0  -0.2794
  7.0    2.6458   49.0   0.6570
  8.0    2.8284   64.0   0.9894
  9.0    3.0000   81.0   0.4121
 10.0    3.1623  100.0  -0.5440
```

Note: the following lines yield the same result

```
>> a = (1:10)' ;
>> A(:,1) = a ;
>> A(:,2) = sqrt(a) ;
>> A(:,3) = a.^2 ;
>> A(:,4) = sin(a) ;
```

Reading-in and writing-out results

1) 'save' and 'load'

```
>> save matlabsession
```

This will create a file "matlabsession.mat"

Then, later, type:

```
>> load matlabsession
```

2) Text files

```
>> samplescript2
>> whos A
  Name      Size                    Bytes  Class
  A        100x10                    8000  double array
Grand total is 1000 elements using 8000 bytes
>> dlmwrite('A.dat',A,'\t')
```

The file "A.dat" can then be read in by Excel, Origin, Sigmaplot, etc.

```
>> data = dlmread('A.dat') ;
```

3) Data stored as images

```
>> data = imread('flash4.jpg','jpg') ;
```

if statements

Along with for loops, another powerful programming logic tool

Assume that all values in the 1-D array **a** should in principle be positive

When a negative value is encountered, you wish to alert the user and set the value to zero

```
>> for i=1:length(a)
  if (a(i) < 0)
    disp([‘Negative value found at index ’,int2str(i)])
    a(i) = 0 ;
  end  % this signifies end of if statement
end % this signifies end of for loop
```

Note 1: There are other, easier ways to test these sorts of things

Note 2: To test whether two things are equal, must use the "double equals" sign

```
if (a(i) == 0)
 disp([‘Zero value found at index ’,int2str(i)])
end
```

MATLAB syntax review

1) The semicolon

Place at the end of a line to suppress output

Also used for vertical concatenation:

```
>> C = [1,4,2 ; 8,0,3] ;
```

2) The colon

To generate vectors of equally-spaced numbers

```
>> a = 0:0.01:5 ;
```

To access all elements along one dimension

```
>> a = A(3,:) ;
```

3) The period

To perform array computations instead of matrix computations

```
>> C = A*B ;
```

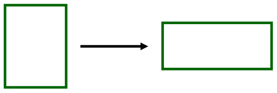
versus

```
>> C = A.*B ;
```

4) The apostrophe

To transpose a matrix

```
>> B = A' ;
```



To delineate a string variable

```
>> a = 'I love matlab' ;
```

Summary

`if` statements, like `for` loops, are powerful computing tools

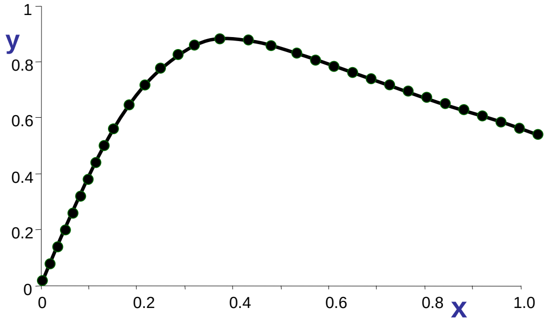
`find` is a useful command for determining not *whether* something is true, but *where* something is true

MATLAB includes several convenient methods for writing out and reading in results

Self-assessment question

A variable `y` is a function of a variable `x` as shown. You wish to determine the values of `y` and `x` at the peak of `y`. You type the following lines into your command window:

```
[max_y,index] = max(y);
disp(['peak y = ',num2str(max_y)])
disp(['x at peak = ', ...
  num2str(index)])
```



Does this give you the correct result?

(A) Yes

(B) No. Instead `index` holds the maximum value and `max_y` determines its location with respect to `x`.

(C) No. You made an error when you extended the third line of code using the ellipsis (…)

(D) No. `max_y` correctly holds the maximum value but `x(index)` is the correct location with respect to `x`.

(E) No. You made an error with the use of the `num2str` function