Data Analysis in Geophysics
ESCI 7205

Bob Smalley
Room 103 in 3892 (long building), x-4929

Tu/Th - 13:00-14:30
CERI MAC (or STUDENT) LAB

Lab – 2, 08/29/13

i) Intro to Matlab.

ii) Intro to Programming.

Start Matlab

Get several windows

Can customize

Do some simple arithmetic –

Use as a calculator

add, multiply, etc. numbers

(also pre-defined [case sensitive]:
sqrt(-1) - i, j
pi
infinity – Inf
Not a Number - NaN)

Standard math operations

+	add
-	subtract
*	multiply
/	divide
^	exponentiate

Result goes into **ans**

Gets "erased" each time you do new calculation.

```
>> 2+3
ans =
     5
>> ans
ans =
     5
>> 5+6
ans =
    11
>> ans
ans =
    11
```

To save it for later, use variable assignment

```
a=3+2
```

Saves result in a "variable" (a place in computer memory, called "**a**", you give it the name – the computer does this, and keeps track of it, for you.)

If you do

```
a=5+2
```

It erases the original a.

NOTE: the "=" here is not a mathematical equals, and the "**a**" is not a mathematical variable

```
a=a+1
```

is perfectly good "math" on the computer

This says -
1) take the value stored in the variable "**a**" (the place in computer memory called "**a**")
2) add one to it
3) and then store the result in the variable "**a**" (the place in computer memory called "**a**")

Also has trig functions

sin

cos

sinh

:

Which take their argument in radians, and

sind

cosd

:

Which take their argument in degrees

Also has inverse trig functions

asin

acos

:

:

With result in radians

asind

acosd

:

:

With result in degrees

And math functions

log
log10
log2

sqrt
exp

...

Make a few vectors by:

Typing them in - `x=[0 .1 .2 … 1]`

Using linspace - `x=linspace(0,1,11)`

Using array & colon notation –

```
x=[start:step:end];
```

or

```
x=start:step:end
```

If you leave the "`:step`" out the step is 1.

End line with "`;`" to supress output

Seeing what's there

```
whos
```

Row vs column vectors (x above is row vector)

```
xc=[0; .1; .2; ... 1]
```

or transpose a row vector to a column vector

```
xc=[0 .1 .2 ... 1]'
xc=x'
```

Try **whos** again and compare.

Arrays (matrices)

Matlab handles vectors same as matrices (actually vectors are matrices to matlab)

Type them in -
As before with vectors but use "**;**" to start new row.

```
x=[1 2 3; 4 5 6]
```

See what you got with **whos**

Predefined arrays

Matlab has a number of predefined arrays

```
ones
zeros
rand
magic
```

etc.

Have to define number rows and columns, single "size", n, gives square nxn matrix

`x = ones(1,5)` row vector

`y = zeros(5,1)` column vector

Can also combine vectors to make arrays

```
a=[1 2 3]
b=[4 5 6]
c=[a b]
d=[a; b]
```

What is difference between **`c`** and **`d`**?
Use **`whos`**.

To get one value from the array you have to give the indices of the location in the matrix (just like math)

```
x(1,3)
```

x better have at least 3 elements

(if x is a vector you can address the elements as x (n), or x(1,n) for row vectors, or x(n,1) for column vectors)

Arithmetic on matrices (vectors)

"standard"

Can add, subtract, multiply (following normal math rules for matrix sizes).

Make some and try it.

Simple arithmetic

If **x** and **y** are vectors of length N you can add them

```
z=x+y
```

Matlab will complain if they are not the same length.

Try it.

Simple arithmetic

You can add or multiply a vector by a constant.

```
z=x+i*y+z0
```

Here I've made a new **complex valued vector**.

Matlab figures out what type of number it is from context.

Matlab handles real and complex numbers.

Simple arithmetic

You can multiply vectors under certain restrictions – they multiply like matrices.

A vector in Matlab is really a matrix with only one row or column, so a vector with N elements is either a 1xN or an Nx1 matrix.

To multiply two matrices the “inner” dimensions have to be the same (Nx1 * 1xN gives an NxN matrix as a result while a 1xN * Nx1 gives a 1x1 [scalar] as a result).

Arithmetic on matrices (vectors)

Matlab extensions

Can multiply, divide, etc. Element-by-element.

Uses dot "." operator.

.*   ./   .^   etc.

Try it.

(add, subtract are already element by element – no .+ and .-)

Reshaping matrices

```
>> a=[1 2 3 4 5 6; 7 8 9 10 11 12]
a =
     1     2     3     4     5     6
     7     8     9    10    11    12
>> a(:)
ans =
     1
     7
     2
     8
     3
     9
     4
    10
     5
    11
     6
    12
```

The colon ":" is a special operator used with arrays in Matlab. Here is says to list all the elements of the (potentially N dimensional) array. It lists them in the order they are stored in memory).

Now reshape the 2x6 matrix **a** into a 3x4 matrix **b**

```
>> b=reshape(a,3,4)
b =
     1     8     4    11
     7     3    10     6
     2     9     5    12
>> b(:)
ans =
     1
     7
     2
     8
     3
     9
     4
    10
     5
    11
     6
    12
```

If you compare b(:) to a(:) you will see they are the same. You have not rearranged the data in memory – just provided a new way to access it (actually a copy of it).

```
>> a(:)
ans =
     1
     7
     2
     8
     3
     9
     4
    10
     5
    11
     6
    12
```

Same but now reshape the 2x6 matrix **a** into a 4x3 matrix **c**

```
>> c=reshape(a,4,3)
c =
     1     3     5
     7     9    11
     2     4     6
     8    10    12
>> c(:)
ans =
     1
     7
     2
     8
     3
     9
     4
    10
     5
    11
     6
    12
>>
```

```
>> a(:)
ans =
     1
     7
     2
     8
     3
     9
     4
    10
     5
    11
     6
    12
```

You can also "cut out" sections of an array using the colon operator

`x(n:m,p:q)`

This will pull elements in rows **n** through **m** and columns **p** through **q** out of the matrix.

```
>> a=[1:16]; b=reshape(a,4,4)
b =
     1     5     9    13
     2     6    10    14
     3     7    11    15
     4     8    12    16
>> c=b(2:3,2:3)
c =
     6    10
     7    11
```

How to store matrix
Two ways

Fortran, Matlab use one – Column Major order
(down the columns, first index varies fastest)

C uses other - Row Major order
(across the rows, last index varies fastest)

Buidling matrix of repeated parts – `repmat`

```
repmat(q,n,m)
```

Takes matrix **`q`** and repeats it **`n`** times as blocks of rows and **`m`** times as blocks of columns.

Try it.
Look and see how stuff stored in memory.

Matlab functions such as sin, cos, etc. work on matrices

sin(x) where x is vector or matrix makes vector or matrix of sines of elements.

Functions are "vectorized"
(should try to write your functions to be vectorized)

In class exercises

1) Plot a formula: `y=x^2`

2) Plot a circle