Lecture 2 for Math 398 Section 952: Numbers, Matrices and Objects

Thomas Shores
Department of Math/Stat
University of Nebraska
Fall 2002

As will be the case with most of these lectures, a good deal of material will come from the text by Desmond Higman and Nicholas Higman, *Matlab Guide*, SIAM, Philadelphia, 2000.

Numbers

Start Matlab. As we indicated in the first lecture, Matlab is a fine calculator. But as with any calculator, one has to be careful:

```
> ((2/7+1000)-1000)-2/7
```

Multiple commands are possible

```
> x=sin(.3),y=cos(.3);x^2+y^2
```

And of course help is always available, specific or general:

```
> help sqrt
> helpwin
```

There is another handy facility that is a word search function:

```
> lookfor elliptic
```

One always has access to the last output:

```
> exp(-3)
> x=ans;
> disp(x)
```

And one can always load and save variables:

```
> save 'junkfile' x y
> % Now clear all variables:
> clear
> who
> load 'junkfile' x
> who
```

Arithmetic

Now a few words about arithmetic, which conforms to double precision IEEE standard. Here eps is the distance from 1 to the next floating point number:

```
> eps
```

```
> 1+eps
> format long
> 1+eps
> format hex
> 1
> 1+eps
> 1+2*eps
> format
```

There are other kinds of numbers. Just for the record:

```
> realmax
> realmin
> realmin/2
> realmax/realmin
> 0/0 % indeterminate...see what Matlab reports
> cos(0)/sin(0)
```

Matrices

This is where Matlab really excels in a way that other computer packages don't. Building and manipulation matrices is quite easy with Matlab:

```
> zeros(2) % create a 2x2 matrix of zeros
> zeros(2,1) % create a 2x1 matrix of zeros
> ones(3)
> ones(3,2)
> eye(3) % create a 3x3 identity matrix
> eye(2,3)
> rand(3)
> A=rand(3)-0.5 % create a 3x3 matrix of random numbers in [-0.5,0.5]

> size(A)
> x=ones(1,5)
> size(x)
> length(x)
```

Of course you can build and reshape matrices from the command line in many ways:

```
> a = [1 2 3
> 4, 5, 7
> 2,4, 6]
> a = [1 2 3; 4, 5, 7; 2,4, 6]
```

One accesses entries or changes them with a standard mathematical notation:

```
> a(1,3)
```

```
> a(1,3) = 10
> a(1,3)
```
One can access a vector (row or column) by a single coordinate:
```
> y = x' % the prime sign ' performs (Hermitian) transpose operation
> x(3)
> y(4) = 7
```
One can build matrices in blocks:
```
> b = [eye(3) a; a eye(3)]
```
The all-important colon notation gets used in two different ways. First as a separator in a type of vector constructor:

> x = 1:5

> y = 1:2:5

> z = (1:0.5:5)'

The other principle use if the colon notation is to work as a wild card of sorts. The colon in a position used for a row indicator means

> c = b(:, 3:5)

Matrix manipulations:
```
> triu(a)
> tril(a)
> diag(a)
> diag(diag(a))
> reshape(a,1,9)
```
Matrix constructions (there is a huge number of special matrices that can be constructed by a special command.) A sampling:
```
> toeplitz((1:5),(1:5).^2)
> hilb(6)
> vander((1:6))
```
Multidimensional arrays: Matlab can deal with arrays that are more than two dimensional. Try the following
```
> a = [1 2; 3 4]
> a(:,:,2) = [1 1;2 2]
> a
```
A final note on matrices: one can even use a convenient Array Editor to modify matrices. Do this: if the Workspace window is not already visible, click on the View button, then check Workspace. Now double-click on the variable 'a', a matrix we created earlier. The Array Editor will open up. Edit a few entries and close the Editor. Confirm your changes by typing at the command line:
```
> a
```

Objects

Objects are instances of classes. For now we're going to confine our attention to some fairly simple types of objects. Matlab has five built-in classes of objects, one of which we've already seen:

- double: double-precision floating poing numeric matrix or array
- sparse: two-dimensional real (or complex) sparse matrix
- char: character structure
- struct: structure array
- cell: cell array

Various matlab toolboxes provide additional class definitions.

Of course, we've seen lots of doubles. Here's another very familiar sort of object, namely a string object:

```
s = 'Hello world'
size(s)
s
disp(s)
```

The struct object is what it sounds like, a way to create structures and access them. There are two ways to build a structure: command line assignments or the struct command. What actually gets constructed is a structure array. Try the following

```
> record.name = 'John Doe'
> record.hwkscore = 372
> record.ssn = [111 22 3333]
> record
> record(2).name = 'Mary Doe';
> record(2).hwkscore = 40;
> record(2).ssn = [222 33 4444];
> record
> record(1).hwkscore
```

Finally, a cell object is an array whose elements can be any other object. For example:

```
> A(1,1) = {[1 2; 3 4]};
> A(1,2) = {'John Smith'}
> A(2,1) = {249}
> A(2,2) = {[1;2;3;4]}
> A{1,2}
> A(1,2)
```