LECTURE 11: ARRAYS, MATRICES AND VECTORS

We saw horizontal row vectors in Section 3.2. Recall the syntax was

```
>> h = [1 2 4 8]
```

The column vector $\mathbf{a} = [2,\ 1,\ 5,\ 9]^T$ can be defined in MATLAB by:

```
>> a = [2; 1; 5; 9]
```

As we saw in Section 3.2, to change the third element to the value 7, type

```
>> a(3) = 7
```

and then check the result by typing

```
>> a
```

Vectors can be defined and initialized in various ways; for example a zero column vector can be set up with the command

```
>> c = zeros(4,1);
```

and then the entries may be input, for example, one at a time. Similar commands include `ones()` and `rand()`.

3.5.1 Matrix definition

Matrices—2D arrays of numbers—can be entered row by row:

```
>> A = [1 2 3; 4 5 6];
```

The entries of a matrix can be accessed by two indices

```
>> i = 1;
>> j = 1;
>> A(i,j)
```

or more directly as in

```
>> A(2,1)
```

Individual entries can be changed as necessary:

```
>> A(1,2) = -1;
>> A
```

MATLAB does not distinguish between matrices and vectors: they are all just *arrays* of various sizes. A MATLAB array can contain symbolic entries but all entries of an array must be the same class.

3.5.2 Rows, columns and submatrices

Suppose that A is a matrix. We can extract the first row of A using:

```
>> A = rand(4,5)
>> h = A(1, :)
```

and you can think of this as "first row, every column" (i.e., the first row of A). Similarly, we get the second column of A using:

```
>> v = A(:, 2)
```

We can also extract a submatrix

```
>> B = A(1:2, 1:3);
```

More generally, the submatrix of rows i to j and columns k to l can be extracted with the command `A(i:j, k:l)`.

3.5.3 Operators on matrices and vectors

As you will be aware, matrix algebra can only be performed on matrices of the right shapes, and we shall suppose that the operations defined below have a meaning; if the matrices are the wrong shape MATLAB will probably complain.

In the following table, A and B are matrices, $\mathbf{u}, \mathbf{v}$ are vectors and s a scalar.

MATLAB command	Function
`s*A`	Scalar multiplication by s
`A*B`	Matrix multiplication of A of B
`A*v`	Multiplication of A by vector v
`A+B`	Matrix addition of A and B (component-wise)
`A^n`	Matrix power
`A.^n`	Component-wise exponentiation
`A .* B`	Component-wise multiplication (A, B same size)
`A ./ B`	Component-wise division
`inv(A)`	Matrix inverse (square matrices)
`x = A \ b`	Solve systems of linear equations $Ax = b$
`A-s`	subtract scalar s from each element
`det(A), trace(A)`	determinant and trace of A
`transpose(A)`	transpose of A
`A'`	conjugate transpose of A
`u' * v`	inner product (dot product) of two *column* vectors
`u * v'`	outer product of two *column* vectors
`cross(u,v)`	cross product of two length-3 vectors
`eye(n)`	$n \times n$ identity matrix
`zeros(n,m)`	$n \times m$ zero matrix
`ones(n,m)`	$n \times m$ matrix of ones
`size(A)`	returns `[n m]` for an $n \times m$ matrix

Here are some examples:

```
>> u = rand(3,1)
>> v = rand(3,1)
>> u' * v
>> u * v'
```

The conjugate transpose and `transpose()` are the same thing for real matrices. MATLAB users tend to mostly use `'`.

In many cases, matrix operators work with symbolic matrices:

```
>> syms x
>> A = sym(round(10*rand(4,4)));
>> I = sym(eye(4,4));
>> B = A - x*I
>> inv(B)
>> pretty(inv(B))
```

The output of the above suggests that inverting matrices is rather hard. Happily, it's not something we have to do very often—for example, this is *not* how MATLAB's "\" solves $Ax = b$.

Exercise 3.11 Let $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, $B = \begin{bmatrix} 1 & 3 & 7 \\ 4 & -5 & 0 \end{bmatrix}$ and $C = \begin{bmatrix} 2 & 5 \\ 4 & 6 \\ -1 & 0 \end{bmatrix}$.

Use MATLAB to find (where possible) $3A$, $A - 2B$, $A + 2C$, AC, CA, AB and A^T. Verify that $(AC)^T = C^T A^T$. □

3.5.4 Simultaneous equations

Exercise 3.12 Consider the set of simultaneous equations

$$3x + y - z = 1,$$
$$5x + y + 2z = 6,$$
$$4x - 2y - 3z = 3.$$

Write this as a system $A\mathbf{x} = \mathbf{b}$, where A is a 3×3 matrix, $\mathbf{x} = [x,\ y,\ z]^T$ and $\mathbf{b}$ is a column vector of length 3. Use the MATLAB backslash \ command to solve the system of simultaneous equations for $\mathbf{x}$. □

3.5.5 Eigenvalues and eigenvectors

The eigenvalue problem for a square matrix A is to find nonzero eigenvectors x and associated eigenvalues λ such that:

$$Ax = \lambda x.$$

(If you haven't seen this before, you will soon in linear algebra—skip this for now and come back later). In MATLAB:

```
>> lambda = eig(A)
>> [V,D] = eig(A)
```

The first form gives a vector of the eigenvalues and the latter gives a diagonal matrix `D` with the eigenvalues on the diagonal. The columns of the matrix `V` contain the eigenvectors.

Exercise 3.13 (a) Use MATLAB to find the eigenvalues and corresponding eigenvectors of the matrix

$$A = \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix}.$$

(b) Now use MATLAB to find the eigenvalues and eigenvectors of

(i) A^3, (ii) A^{-1}, (iii) $A - 6I$ (where I is the identity), (iv) $(A + 3I)^{-1}$.

In each case, can you spot the relationship of the eigenvalues and eigenvectors found in (b) to those found in (a)? □

3.6 Meshgrids and 3D plots

We looked at indexing into vectors in Section 3.2.2. Understanding this enables concise code, particularly for plotting.

3.6.1 Logical masks

Let's reconsider an example from earlier using the `find` command:

```
>> L = 2:.1:4;
>> % approach 1
>> I = find(L > 3.05);
>> L(I)
>> % approach 2
>> B = L > 3.05;
>> L(B)
```

The results of `L(I)` and `L(B)` are the same. But compare `I` and `B`. Note that `I` contains just the indices of those elements which are greater than $\frac{3}{10}$. On the other hand, `B` contains 1's where the condition is true and 0's where it is false. `B` is known as a "logical mask".

Here's an example of what can be done with this sort of masking

```
>> x = linspace(-1, 1, 512);
>> y = cos(2*pi*x);
>> y(y > 0) = 0;
>> y(x > 0.5) = 2;
>> plot(x, y, 'b.-');
```

Exercise 3.14 Explain what each line of the previous job accomplishes. Change "512" in the above code to "20" and make the code explicitly display the two logical masks. If you swap the order of the two "`y(...) = ...`" lines, does the picture change? Why or why not? □

3.6.2 Meshgrid

We can build special 2D arrays (matrices) of x and y coordinates for use in plotting:

```
>> x = linspace(-2,2,60);
>> y = linspace(-1,1,40);
>> [xx,yy] = meshgrid(x,y);
>> f = sin(2*pi*xx) .* cos(pi*yy);
>>
>> figure(1); clf;
>> surf(xx, yy, f);
>> axis equal
>> xlabel('x'); ylabel('y'); zlabel('f');
>>
>> figure(2); clf;
>> pcolor(xx, yy, f);
>> axis equal; axis tight;
>> xlabel('x'); ylabel('y');
>> colorbar
```

Meshgrid is essentially the Cartesian product of the 1D grids in the x and y directions. Examine the following output to understand what meshgrid does:

```
>> x = linspace(-2, 2, 5)
>> y = linspace(-1, 1, 4)
>> [xx, yy] = meshgrid(x, y)
```

3.6.3 3D plotting

Exercise 3.15 In the plotting code above, try adding "`shading interp`", "`shading flat`", or "`shading faceted`" after each plot.

After the `surf` command, try adding these command one after another and examining what each one does: "`shading interp`", "`camlight left`", "`lighting phong`", "`material shiny`".

Use your mouse and the rotate and zoom tools to interact with your figures. □

Next let's combine meshgrids with logical masks. This gives a way to modify regions for plotting.

```
>> x = linspace(-2,2,60);
>> y = linspace(-1,1,40);
>> [xx,yy] = meshgrid(x,y);
>> f = sin(2*pi*xx) .* cos(pi*yy);
>> I = (xx > 0)  &  (yy > 0);
>> f(I) = 0;   % or try: f(I) = f(I)+1;
>>
>> figure(1); clf;
>> surf(xx, yy, f);
>> axis equal
>> xlabel('x'); ylabel('y'); zlabel('f');
>>
>> figure(2); clf;
>> pcolor(xx, yy, f);
>> axis equal; axis tight;
>> xlabel('x'); ylabel('y');
>> colorbar
```

Exercise 3.16 Modify the above code so that for every point where f is negative, the value is replaced with three times its absolute value.

Exercise 3.17 Modify the above code to plot $f = 3-(x^2+y^2)$ in the interior of the following triangle. Set $f = 0$ outside the triangle.

The interior of the triangle is defined by the inequalities

$$y < 0.25x + 0.5, \quad x < 1.3 + 0.15y, \quad \text{and} \quad y > -0.4x - 0.5.$$

Also, try setting the outside values to `inf` and `nan` (not a number). □

Exercise 3.18 What does this code do?

```
>> A = rand(10,10);
>> I = A > 0.8;
>> A(~I) = 0;
```

□

Exercise 3.19 Plot the function `besselj(nu,r)` where $r = \sqrt{x^2+y^2}$ on a circle of radius 20 for various integer values of ν. Perhaps animate your plots over $\nu = 0, 1, 2, \ldots$ using the `pause` command. □