Inf1-CG Lab 3, Week 4
Matrices

Peter Bell

January 30, 2015

This practical uses Matlab. Before attempting to do the exercises read the Matlab slides made available from the course web site (`http://www.inf.ed.ac.uk/teaching/courses/inf1-cg/labs.html`). The site also provides slides introducing basic operations on matrices and their algebraic properties.

1 Introducing Vectorisation

One of the most powerful features of Matlab is the ability of most standard numerical functions to operate on vectors and matrices as well as scalars. Computations that would normally be performed using loops in more traditional programming languages can often be carried out in Matlab using a single command, acting on a vector or matrix. The process of converting code to take advantage of this functionality is known as *vectorisation*. As well it often being simpler to write vectorised code, it will also run faster, as Matlab has been specially optimised for this kind of computation.

An example of a function that works with vectors or matrices just as well as scalars is `max`. For example:

```
max(3,4) = 4                    % operating on two scalars
max([ 1 3 4 -2 2 ]) = 4         % operating on one vector
max([ 1 2  4
```

```
      3 5  1
      7 1 -2 ]) = [ 7 5 4 ]     % operating on one matrix
                                % (a second call to max would find
                                % the overall maximum value)
```

Even comparison operators can be vectorised. For example, if

```
M = [ 1 2  4
      3 5  1
      7 1 -2 ]
```

then the value of `M==1` is another matrix:

```
M = [ 1 0 0
      0 0 1
      0 1 0 ]
```

Connected with vectorisation is the ability in Matlab to refer to elements of vectors or matrices using vectors as the subscripts. For example suppose we have `a = [ 1 3 ]`. Then

```
M(a, a) = [ 1  4
            7 -2 ]
```

This can be extremely useful!

In this practical you will use vectorisation techniques to construct or compute a variety of vectors and matrices, **without using any loops**. In most cases the answers can be obtained using a single line of code! Everything you use should be generaliseable to vectors and matrices of any length.

All the vectors and matrices referred to in the questions can be loaded into Matlab from the file `lab3_data.mat`.

2 Manipulating vectors

Throughout the practical, it will be useful to remember the notation `start:step:end` gives a vector with first element `start`, successive elements increasing by `step`, with final element `end`. Also, the `(:)` notation gives all the elements in a particular dimension (see slides on Matlab matrices for more details).

Consider the vector **a**:

```
a = [ 2 -3  -4  1  0  -2  3  1  2 ]
```

From **a**, obtain each of the following, using one line of Matlab code in each case:

1. **a**, but with every third element removed.
2. **a**, but with every third element replaced by the corresponding multiple of 3
3. **a**, but with every negative element replaced by zero
4. **a**, but with every negative element multiplied by -2
5. A count of the number of positive elements of **a**.
6. The outer product of **a**.

3 Manipulating Matrices

We will now use matrix **M**, given by:

```
M =
    [ -4     5     3     0     3    -5    -1     3    -1     0
      4    -3     4    -1    -3    -2     3     3     2    -3
      1    -4     5     2    -1    -1     3    -3     3     0
      0    -4    -3     2     2    -2    -1     4     3    -4
      1    -4    -4     0    -1    -3    -3     5    -2    -4
      0    -1     2    -2     2     3     3     0     0     4
```

```
 1    -1    -4    -4    -1    -1     4     4    -4     1
 0    -1     0     1     2     4    -2     1    -4     4
 2     3     0    -2     2    -1     2    -3    -4     2
 0     1     4    -5    -1     3    -1    -3     2     1 ]
```

In this section the functions `zero`, `ones`, `diag` and `eye` will be useful. Read the Matlab slides provided on the course web site:

For **M**, obtain the following:

1. **M**, but with every second row removed
2. **M**, but with columns 5 and 10 set to zeros
3. **M**, but with zeros down the (leading) diagonal
4. **M**, but with fives down the diagonal
5. **M**, but with every diagonal element multiplied by 2
6. A matrix consisting of the *lower triangular* elements of **M** (use the built-in Matlab function)
7. A *symmetric* matrix, with the lower triangular elements matching the lower triangular elements of **M**.

Also, compute:

8. a column vector containing the maximum element from each *row* of **M**.
9. the sum of all the rows of **M** which have a positive diagonal element

4 Matrix Operations

We have the following four matrices, operating on 2-dimensional vectors:

```
M1 = [  0 1    M2 = [ 0 1    M3 = [ 3 0    M4 = [ 0 0
       -1 0 ]          1 0 ]          0 1 ]          0 1 ]
```

1. For each of these matrices, describe geometrically what action (*transformation*) it performs. To do this, experiment with matrix-vector multiplication in Matlab, using a variety of different vectors. It will be helpful to display the original vectors and the transformed vectors on the same plot using the `plotv` function from Practical 1.

2. Consider the matrix given by `M2*M1*M2`. What action does it perform? Can you explain why?

3. Construct a 4-dimensional matrix (operating on 4-dimensional vectors) to set the 2nd and 4th element of a vector to zero.

4. *(Optional)* `M4` is an example of a *projection*. Construct a matrix that projects a 2-dimensional vector onto the line $y = x$, rather than onto one of the axes.