Math 451 - Matlab Lesson #2:
Matrices

1 Matrix Arithmetic

The name 'Matlab' is short for 'Matrix Laboratory' since the fundamental object in Matlab is the matrix. Even the simple numbers we considered in the last section are, for Matlab, just 1×1 matrices.

So what is a matrix? A matrix is a block of numbers (called *elements*), organized into rows and columns. The *dimensions* of the matrix, $n \times m$, are the number of rows (n) and the number of columns (m).

Example 1.1:

$$\begin{bmatrix} 2 & 8 & 7 & 2 \\ 1 & -5 & 15 & -3 \\ -4 & 11 & 7 & 9 \end{bmatrix}$$

This is a 3×4 matrix.

Matrices of the same dimension can be added simply by adding the corresponding entries.

Example 1.2:

$$\begin{bmatrix} 9 & 5 & 1 \\ 11 & -7 & 8 \end{bmatrix} + \begin{bmatrix} -2 & 5 & 7 \\ 12 & 6 & -8 \end{bmatrix} = \begin{bmatrix} 7 & 10 & 8 \\ 23 & -1 & 0 \end{bmatrix}$$

Matrices of different dimensions cannot be added.

Example 1.3:

$$\begin{bmatrix} 4 & -9 & 3 \end{bmatrix} + \begin{bmatrix} 5 \\ -8 \\ 3 \end{bmatrix} = \text{Not defined}$$

Exercise 1.1: Add the matrices below, and give the dimension of the result.

$$\begin{bmatrix} 9 & 5 \\ 10 & 8 \\ -6 & -2 \end{bmatrix} + \begin{bmatrix} -7 & 5 \\ 3 & -8 \\ 5 & 4 \end{bmatrix} =$$

Dimension = _____

Matrices of the correct dimensions may also be multiplied, but the process is much more complicated. The number of columns in the left matrix must be equal to the number of

rows in the right matrix. The resulting matrix will have the same number of rows as the left matrix, and the same number of columns as the right matrix.

Example 1.4: Multiplying a 4×2 matrix by a 2×3 matrix is defined ($2 = 2$). The result is a 4×3 matrix.

Note that the product in the reverse order (2×3 times 4×2) is **not defined** since $3 \neq 4$.

In a product of two matrices, to find the number in the n-th row and m-th column take the n-th row of the left matrix and the m-th column of the right matrix, multiply corresponding elements, and then sum these products.

Example 1.5: Find the entry in the first row, first column for the product of the two matrices below.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} ??? & \\ & \end{bmatrix}$$

First note that we're multiplying a 2×3 by a 3×2, so the multiplication is defined, and the resulting matrix will be 2×2.

The first row on the left is $1 \quad 2 \quad 3$, the first column on the right is $7 \quad 9 \quad 11$. Multiplying and adding gives us:

$$1 \cdot 7 + 2 \cdot 9 + 3 \cdot 11 = 7 + 18 + 33 = 58$$

Exercise 1.2: Calculate the other three entries in the product matrix in the example above.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & \\ & \end{bmatrix}$$

Exercise 1.3: If it is defined calculate the product of the matrices below.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 10 \\ 11 \\ 12 \end{bmatrix} =$$

2 Matrix Arithmetic in Matlab

The easiest way to enter a relatively small, simple matrix into Matlab is to type in the entries between `[` and `]` signs. The entries should be separated by spaces, and the rows by `;` 's.

Example 2.1: Give the name `A` to the matrix

$$\begin{bmatrix} 2 & 8 & 7 & 2 \\ 1 & -5 & 15 & -3 \\ -4 & 11 & 7 & 9 \end{bmatrix}$$

```
>> A = [2 8 7 2; 1 -5 15 -3; -4 11 7 9]
  A =  2   8   7   2
       1  -5  15  -3
      -4  11   7   9
```

The Matlab command `size( )` returns the dimensions of the matrix that it is applied to.

Example 2.2:

```
>> size(A)
ans =   3   4
```

since A is a 3×4 dimensional matrix.

Exercise 2.1: What would you write in Matlab to give the name `B` to the following matrix?

$$\begin{bmatrix} 9 & 5 & 1 \\ 11 & -7 & 8 \end{bmatrix}$$

```
>> _____
```

Exercise 2.2: What would you write in Matlab to give the name `C` to the following matrix?

$$\begin{bmatrix} 10 \\ 11 \\ 12 \end{bmatrix}$$

```
>> _____
```

It is also possible to make bigger matrices from smaller ones using the same notation, but entering matrices instead of numbers. The only restriction is that the result must be a rectangular block of numbers (that is, each row must have the same number of columns and vice versa).

Example 2.3:

```
>> [A C]
ans =  2   8   7   2  10
       1  -5  15  -3  11
      -4  11   7   9  12
```

Exercise 2.3: What would you write in Matlab to use the matrices `B` and $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ to make the matrix:

$$\begin{bmatrix} 1 & 2 & 3 \\ 9 & 5 & 1 \\ 11 & -7 & 8 \end{bmatrix}$$

```
>> _____
```

As with numbers, Matlab makes it easy to do matrix arithmetic—as long as the operation is defined!

Example 2.4:

```
>> B*A

 ans = 19   58  145   12
      -17  211   28  115
```

Example 2.5:

```
>> A + B

??? Error using ==> plus
Matrix dimensions must agree.
```

Exercise 2.4: Of the expressions below decide **in your head** which quantities are defined and circle them. For those that are defined use Matlab to find the result.

```
A*B   A*C     C*A     B*C
C*B   A + C   B + C   C + C
```

Results:

Another important operation that can be performed on matrices is *transposition*, or exchanging the rows and columns of a matrix. In Matlab this is accomplished with the apostrophe `'`.

Example 2.6:

```
>> B'

ans = 9  11
      5  -7
      1   8
```

Exercise 2.5: Find the transpose of the matrix `C`

In Matlab the apostrophe is also used for *complex conjugation*, so if you apply it to a matrix that has complex elements you will get the *transpose-conjugate* matrix.

Example 2.7:

```
>> D = [1+i 2+3*i]

D = 1.0000 + 1.0000i   2.0000 + 3.0000i

>> D'

ans = 1.0000 - 1.0000i
      2.0000 - 3.0000i
```

3 Arrays

A $1 \times n$ dimensional matrix is often called a list or an *array*. Arrays are very important in Matlab, so there are special methods for generating them. One method is to use the form:

[<begin>:<step>:<end>]

This generates an array whose first element is <begin>, whose second is <begin> + <step>, third is <begin> + 2<step>, etcetera until <end> is reached or exceeded.

Example 3.1:

```
>> K = [3 : 0.5 : 5]

K = 3.0000   3.5000   4.0000  4.5000   5.0000
```

Example 3.2:

```
>> L = [6 : -2 : -3]

L = 6   4   2   0  -2
```

Example 3.3: How many elements are in the array [15 20 25 ... 95 100]?

```
>> M = [15 : 5 : 100];
>> size(M)

ans =   1  18
```

There are 18 elements in the array.

If the middle quantity <step> is omitted, then Matlab assumes it is +1.

Example 3.4:

```
>> [3:7]

ans =   3   4   5   6   7
```

A second method of generating arrays has the form:

`linspace(` <begin>, <end>, <length> `)`

This generates an array of length <length> whose first element is <begin>, and whose **last** is <end>. The intermediate elements are spaced evenly between <begin> and <end>.

Example 3.5:

```
>> K = linspace(3, 5, 5)

K = 3.0000   3.5000   4.0000  4.5000   5.0000
```

Exercise 3.1: How would you use `linspace` to define `L` (as in **example 3.2**)?

```
>> L = linspace(       ,       ,       )
```

Exercise 3.2: What array would the command `linspace(3, 9, 4)` produce?
(Try to figure it out in your head, then check using Matlab.)

```
ans =
```

4 Elements and Element-wise operations

Up until now we have referred to matrices as single objects, but what about the elements? How do you deal with the numbers that make up the matrix on an individual basis?

If you want to refer to or change one element in a matrix you put the matrix name, then the row and column of the element in parentheses. If the matrix is one-dimensional (only one row or column) you need only put one number in parentheses.

Example 4.1:

```
>> A(2,3)

A(2,3) = 15
```

Example 4.2:

```
>> C(2) = 7;
>> C

C = 10
     7
    12
```

Exercise 4.1: What would you write in Matlab to change the element in the first column, second row of `B` to zero?

```
>> ____________________________
ans =   9   5   1
        0  -7   8
```

The colon operator (`:`) may be used as 'the entire row' or 'the entire column'.

Example 4.3:

```
>> A(2,:)

ans = 1  -5  15  -3
```

Exercise 4.2: What would you write in Matlab to produce the second column in `B`?

```
>> ____________________________
ans =  5
      -7
```

Of course you can also use an array as indexes as long as the elements are positive integers.

Example 4.4:

```
>> A(1:2,2:3)

ans =   8   7
       -5  15
```

Exercise 4.3: What would you write in Matlab to produce the following sub-matrix of `A`?

```
>> A(     ,     )

ans =   1  -5  15
       -4  11   7
```

We have seen how matrix addition and subtraction simply add or subtract the corresponding elements of two matrices of the same dimension. What if we wanted to multiply corresponding elements? Recall that matrix multiplication **does not** do this. However element-wise multiplication will.

A period (`.`) followed by any operation makes the operation *element-wise*.

Example 4.5: The following matrix operation is not defined since you cannot multiply a 3×1 matrix by another 3×1 matrix.

```
>> C*C

??? Error using ==> mtimes
Inner matrix dimensions must agree
```

However with element-wise multiplication

Example 4.6:

```
>> C.*C

ans = 100
       49
      144
```

Similarly `C^2` is undefined since `C^2` = `C*C`, but

Example 4.7:

```
>> C.^2

ans = 100
       49
      144
```

Example 4.8:

```
>> E = [1 2; 3 4];
>> F = [5 6; 12 8];
>> F./E

ans = 5   3
      4   2
```

but notice that

```
>> F/E

ans =  -1   2
      -12   8
```

hence matrix right division (which we haven't yet discussed) is different from element-wise right division.

Exercise 4.4: What will `E.*F` be? What will `E./F` be?
(Try to do them in your head first, then check with Matlab.)

Let's consider left division. Element-wise it's the same as right division.

Example 4.9:

```
>> E.\F

ans = 5   3
      4   2
```

since you're still dividing the elements of `F` by the elements of `E`. However for matrix left division

Example 4.10:

```
>> E\F

ans =  2.0000  -4.0000
       1.5000   5.0000
```

the result is different. We'll discuss this more in the next section.

Like operations, some functions treat a matrix as a matrix while others act element-wise. In general those functions which don't deal with a matrix property directly act element-wise. This includes all the familiar functions of one variable like `sin`, `log`, etc.

Example 4.11:

```
>> G = pi.*[0: 0.5: 2]

G = 0   1.5708   3.1416   4.7124   6.2832

>> sin(G)

ans = 0   1.0000   0.0000  -1.0000  -0.0000
```

We will use this property a great deal when we get to graphing functions (Lesson 5).

5 Matrix Division

Matrices have a wild variety of applications, but certainly they are most useful for solving systems of linear equations. The first step in doing so is to put such a system in the form of a matrix equation.

Example 5.1: The following system of three linear equations and three unknowns can be put into matrix form.

$$\begin{aligned} 4x - 3y + 2z &= 7 \\ 2x + 5y + z &= -1 \\ -x + 7y - 6z &= 4 \end{aligned}$$

First we consider it as an equation of two 3×1 matrices,

$$\begin{bmatrix} 4x - 3y + 2z \\ 2x + 5y + z \\ -x + 7y - 6z \end{bmatrix} = \begin{bmatrix} 7 \\ -1 \\ 4 \end{bmatrix}$$

then we write the left side as a product of a 3×3 and a 3×1

$$\begin{bmatrix} 4 & -3 & 2 \\ 2 & 5 & 1 \\ -1 & 7 & -6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 7 \\ -1 \\ 4 \end{bmatrix}$$

Exercise 5.1: Put the following system of four linear equations and four unknowns into matrix form.

$$\begin{aligned} x_1 + 5x_2 - 2x_3 + 4x_4 &= 1 \\ -3x_1 + 6x_2 + 9x_3 - 7x_4 &= 2 \\ 4x_1 + 7x_2 + 3x_3 + 3x_4 &= -2 \\ 5x_1 - 6x_2 + x_3 + x_4 &= -1 \end{aligned}$$

How then do we solve such systems? First we put the system into the form: $AX = B$ where A is an $n \times n$ matrix that we know, B is an $n \times 1$ matrix that we know, and X is an $n \times 1$ matrix that we don't know. To solve for X we need only **left**-divide both sides by A. Thus,

$$AX = B \quad \Rightarrow \quad A\backslash AX = A\backslash B \quad \Rightarrow \quad X = A\backslash B$$

Example 5.2: Solve the system

$$\begin{aligned} 4x - 3y + 2z &= 7 \\ 2x + 5y + z &= -1 \\ -x + 7y - 6z &= 4 \end{aligned} \quad \Longleftrightarrow \quad \begin{bmatrix} 4 & -3 & 2 \\ 2 & 5 & 1 \\ -1 & 7 & -6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 7 \\ -1 \\ 4 \end{bmatrix}$$

```
>> A = [4 -3 2; 2 5 1; -1 7 -6];
>> B = [7; -1; 4];
>> X = A\B
X =   2.1469
     -0.6923
     -1.8322
```

Therefore $x \approx 2.1469$, $y \approx -0.6923$, and $z \approx -1.8322$. It's easy enough to check our answer, just recall that $x =$ `X(1)`, $y =$ `X(2)`, etc.

Example 5.3: Check the first equation,

```
>> 4*X(1) - 3*X(2) +2*X(3)
ans =   7.0000
```

or check the whole thing.

```
>> A*X
ans =   7.0000        -1.0000         4.0000
```

Exercise 5.2: Solve the system:

$$\begin{aligned} x_1 + 5x_2 - 2x_3 + 4x_4 &= 1 \\ -3x_1 + 6x_2 + 9x_3 - 7x_4 &= 2 \\ 4x_1 + 7x_2 + 3x_3 + 3x_4 &= -2 \\ 5x_1 - 6x_2 + x_3 + x_4 &= -1 \end{aligned}$$

$x_1 =$ _____
$x_2 =$ _____
$x_3 =$ _____
$x_4 =$ _____

Warning: You can divide by any number except zero, but matrices are trickier. First, you may only divide by square matrices. Second, you may only divide by matrices whose *determinant* is not zero. Moreover if the determinant of a matrix is very small then division by that matrix may lead to large numerical errors.

Example 5.4: Matlab can easily calculate determinants.

```
>> det(A)
ans = -143
```

So you may (and did) divide by the matrix `A`.

```
>> C = [1 2; 2 4];
>> det(C)
ans = 0
```

So you **cannot** divide by the matrix `C`. If you try,

```
>> D = [5 7];
>> D/C
Warning: Matrix is singular to working precision.
ans = -inf   inf
```

6 Standard Matrices

Matlab has functions which generate certain standard matrices in whatever shape is wanted.

Example 6.1:

```
>> zeros(2,3)

ans =   0   0   0
        0   0   0

>> ones(3,5)

ans =   1   1   1   1   1
        1   1   1   1   1
        1   1   1   1   1

>> I = eye(3)

I =   1   0   0
      0   1   0
      0   0   1
```

This last is called the *identity matrix*. It's called this because it is the multiplicative identity for matrix multiplication. That is,

Example 6.2:

```
>> I*A

ans = 4  -3   2
      2   5   1
     -1   7  -6
```

Note that this is different from the element-wise identity, `ones`.

Example 6.3:

```
>> I.*A

ans = 4   0   0
      0   5   0
      0   0  -6

>> id = ones(3)

id =   1   1   1
       1   1   1
       1   1   1

>> id.*A

ans = 4  -3   2
      2   5   1
     -1   7  -6
```

Note as well that `zeros`, `ones`, `eye`, generate a square matrix if given only one dimension.

Exercise 6.1: What would you write in Matlab to give the name `E` to the following matrix?

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

>> _____

Exercise 6.2: What would you write in Matlab to give the name `F` to the following matrix? (See **exercise 2.3**)

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

>> _____