Introduction to Modelling

1. Introduction to MATLAB

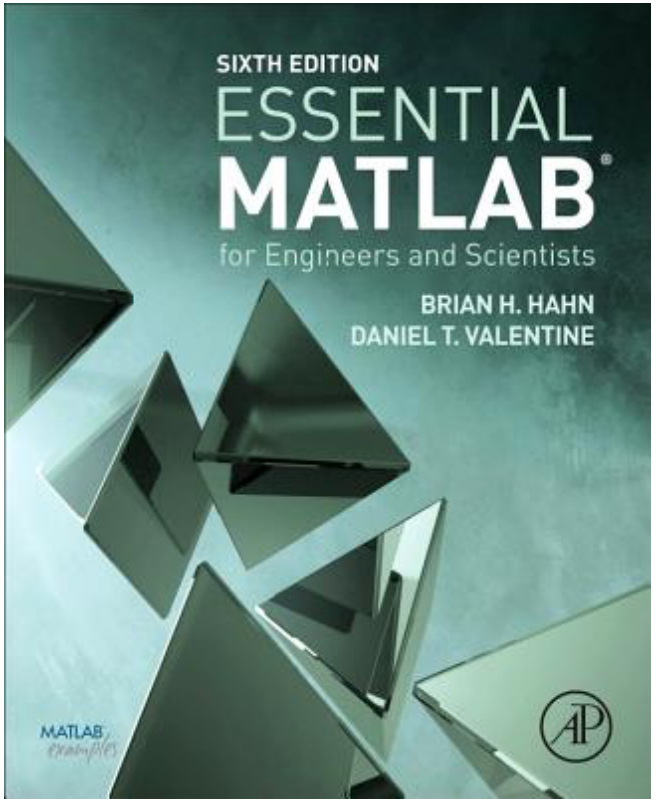
Dr. Jim Duggan,
School of Engineering & Informatics
National University of Ireland Galway.
https://github.com/JimDuggan/CT248

Course Overview

- Lectures
  - 5-6 Monday (IT125)
  - 12-1 Wednesday (Larmor SC002)
- Labs (starting week 2)
  - IT106
  - Continuous Assessment 30% of course mark
- 8 Lab Assignments
- MATLAB & Simulink

MATLAB

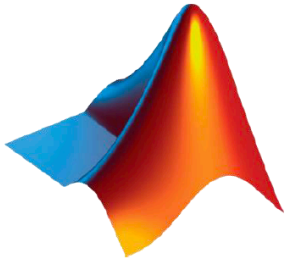
- MATLAB is a powerful technical computing system for handling scientific and engineering calculations.
- MATrix LABoratory
- Designed to make matrix computations particularly easy



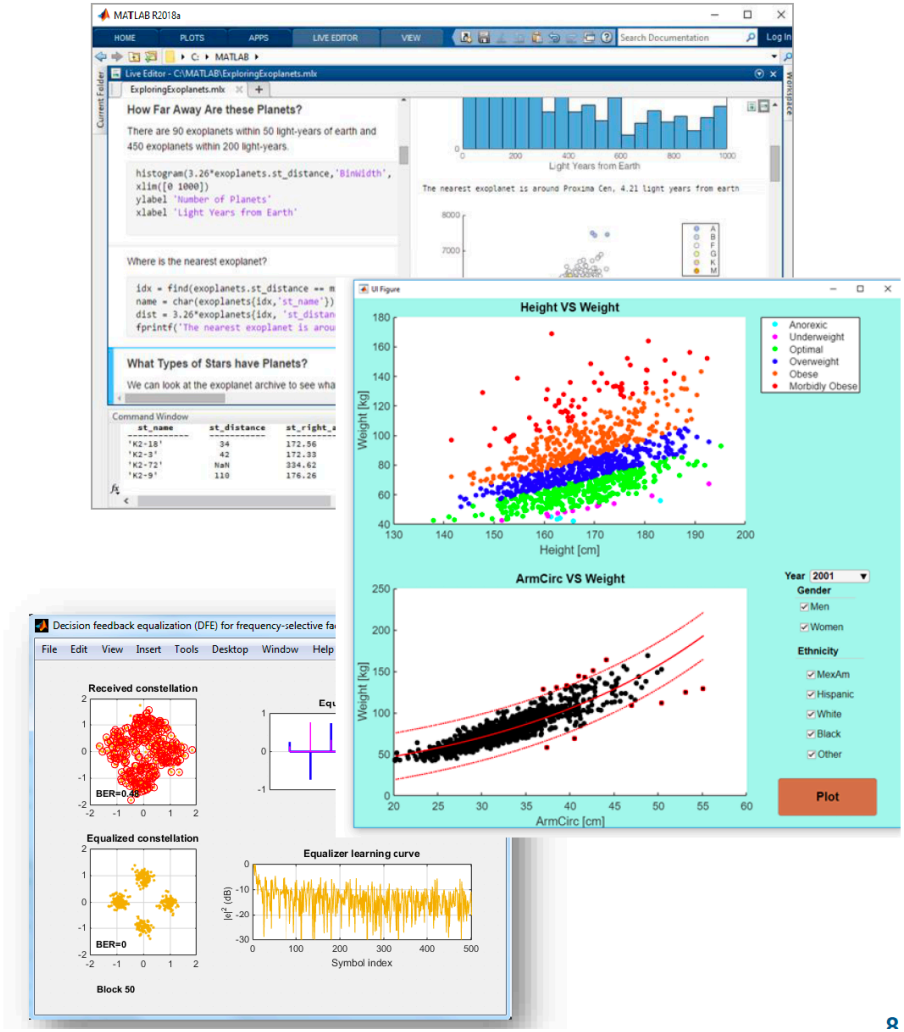
Hahn and Valentine, "Essential MATLAB for Engineers and Scientists", 6th Edition.

MATLAB

What is MATLAB?



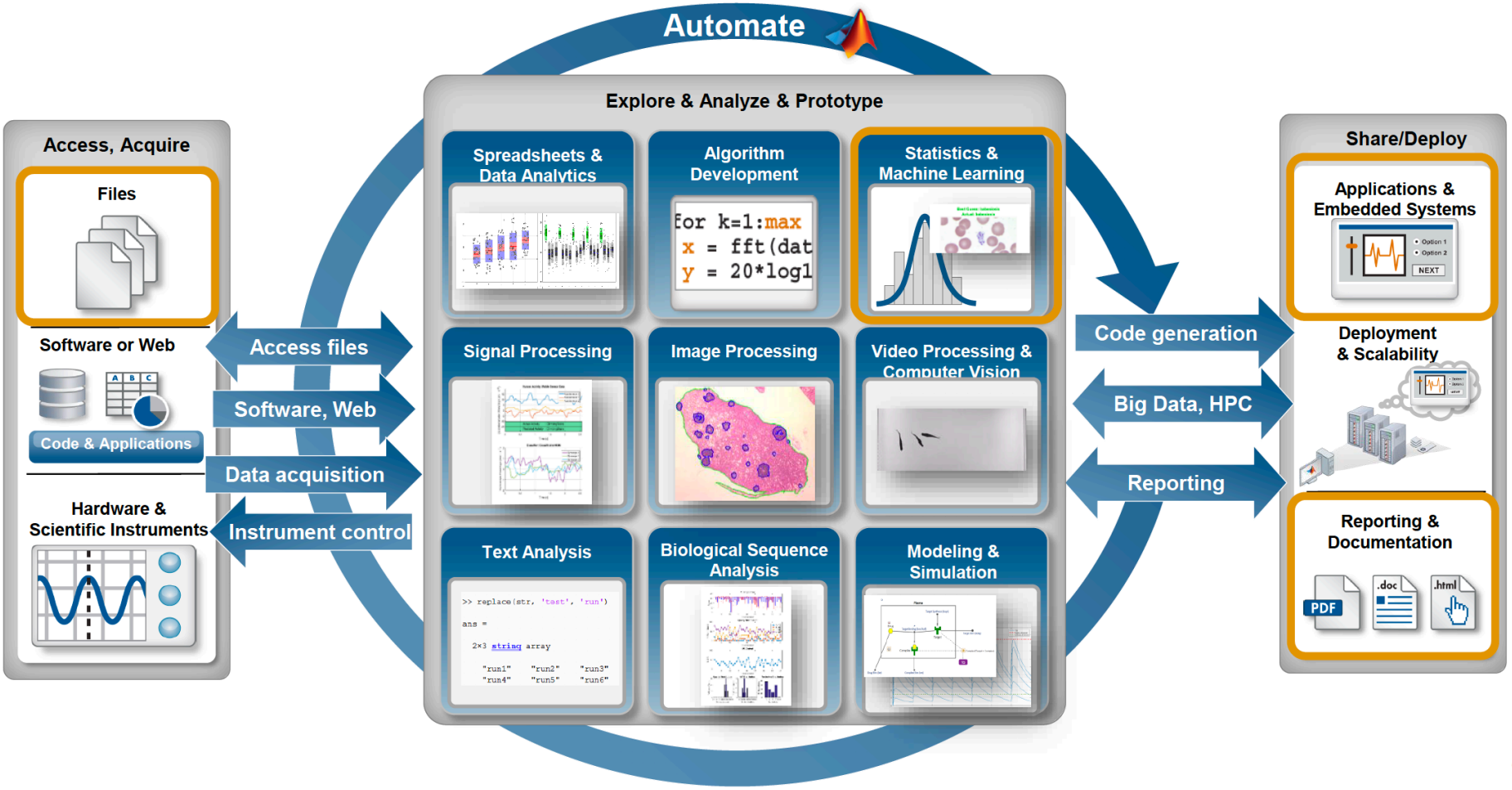
- High-level language
- Interactive development environment
- Used for:
  - Numerical computation
  - Data analysis and visualization
  - Algorithm development and programming
  - Application development and deployment



MATLAB

MathWorks

Computing Technical Workflow



9

Introduction

- Variables, operators and expressions
- Basic input and output
- Repetition (for)
- Relational Operators and Decisions (if)
- Arrays (including vectors and Matrices)

(1) Variables

- Variables are fundamental to programming
- A variable name must comply with two rules:
  - It may consist only of letters a-z, digits 0-9, and the underscore (_)
  - It must start with a letter
- MATLAB only remembers the first 63 characters

Case Sensitivity

- MATLAB is case-sensitive
- Command and function names are case sensitive
- “Camel caps” or underscores can be used for longer names
- The semi-colon prevents the variable value from being displayed

```
carSpeed  = 20;

car_speed = 20;
```

https://sites.google.com/site/matlabstyleguidelines/naming-conventions

The MATLAB Workspace

- All the variables created during a session remain in the workspace until you *clear* them.
- The command *who* lists all the names in your workspace
- The command *whos* lists the size of each variable

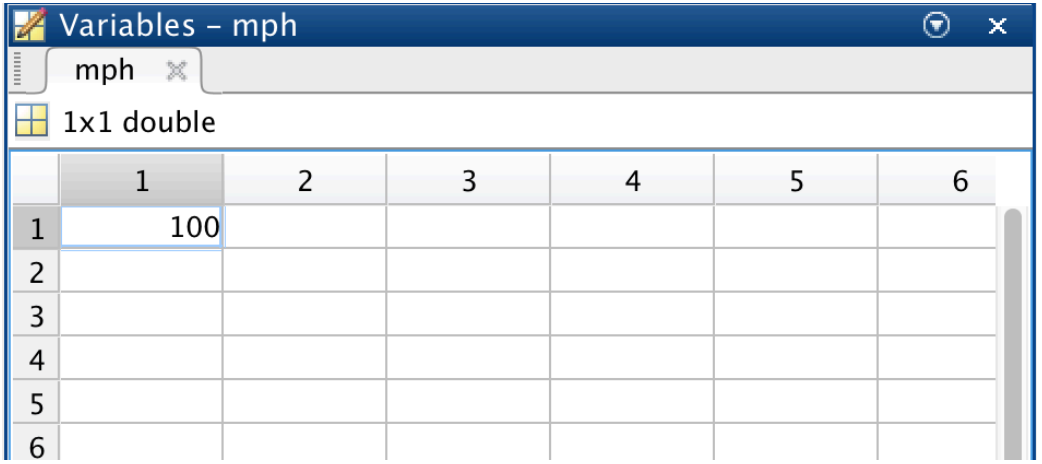
```
clear;

mph  = input ('Please enter the
speed in mph : ');

kph = mph * 1.6;

fprintf("%d mph = %d
kph\n",mph,kph);
```

```
>> whos
  Name     Size          Bytes  Class     Attributes

  kph      1x1               8  double
  mph      1x1               8  double
```

Explanation

- Each variable occupies 8 bytes of storage
- 1x1 is a scalar
- The Workspace browser on the desktop provides a useful visual representation of the workspace
- Workspace values can be changed using the array editor

```
>> whos
  Name      Size            Bytes  Class     Attributes

  kph       1x1                 8  double
  mph       1x1                 8  double
```



Expressions

- An expression is a formula consisting of variables, numbers, operators and function names
- It is evaluated when you enter it at the MATLAB prompt
- Note that MATLAB uses the function ans to return the last expression to be evaluated but not assigned to a variable.

```
>> 2 * pi

ans =

   6.2832

>>
>> ans

ans =

   6.2832
```

Statements

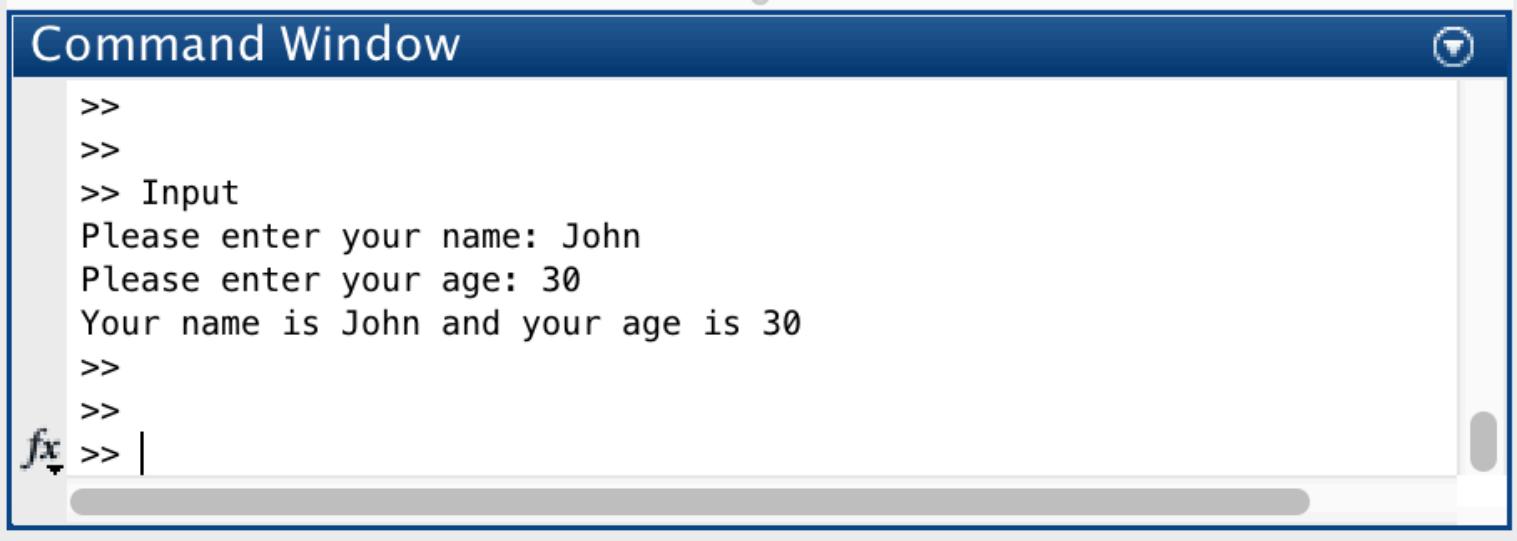
- MATLAB statements are frequently of the form

  variable = expression

  s = u*t – g/2 * t.^2
- This is an example of an assignment statement, with the value of the expression assigned to the variable on the LHS
- A semi-colon at the end suppresses output
- Statements that are too long for a line can be extended with an ellipsis of at least three dots
- Statements on the same line can be separated by commas or semicolons.

(2) Input and Output

```
clear;

% 's' forces the input to be a character string
name = input ('Please enter your name: ', 's');

age  = input ('Please enter your age: ');

fprintf("Your name is %s and your age is %d\n",name,age);
```



Output: the disp statement

- The general form of disp for a numeric variable is

  disp(variable)
- To display more than one variable, embed variables into an array
- All components of a MATLAB array must be the same type.

```
>> disp('Hello world')
Hello world
>> disp('Hello ''world''')
Hello 'world'
>>
>> x = 2

x =

     2.00

>> x = 2;
>>
>> disp(['The answer is ', num2str(x)])
The answer is 2
>>
>> disp([2 3 x]);
     2.00     3.00     2.00
```

Convert F to C
Input – Process - Output

```
% Script file for converting temperatures from F to C
% Taken from Essential MATLAB (D. Valentine) p 42

% Step 1: Get the input
 F = input ('Enter the temperature in degrees F: ');

% Step 2: Convert to C
C = (F - 32) * 5 / 9;

% Step 3: Display the result
fprintf("The temperature of %f (F) is %f (C)\n",F,C);
```

Arithmetic Operators

- The evaluation of an expression is achieved by means of arithmetic operators
- The arithmetic operations on two scalar constants /variables is shown
- Left division seems curious, however matrix left division has an entirely different meaning

Operation	Algebraic Form	MATLAB
Addition	a + b	a + b
Subtraction	a - b	a - b
Multiplication	a x b	a * b
Right Division	a/b	a / b
Left Division	b \ a	b \ a
Power	a^b	a ^ b

Precedence	Operator
1	Parentheses (round brackets)
2	Power, left to right
3	Multiplication & Division, left to right
4	Addition and Subtraction, left to right

Challenge 1.1

Evaluate each of these expressions.

```
a1 = 1 + 2 * 3;

a2 = 4 / 2 * 2;

a3 = 1 + 2 / 4;

a4 = 1 + 2 \ 4;

a5 = 2 * 2 ^ 3;

a6 = 2 * 3 \ 3;

a7 = 2 ^ (1+2)/3;
```

(3) Repetition – for statement

```
for i = 1:5
    disp(i);
end
```

```
1.00

2.00

3.00

4.00

5.00
```

The for loop repeats statements a specific number of times, starting with i = a and ending with i = b, incrementing i by 1 each iteration of the loop.

The number of iterations will be b - a + 1.

The basic for construct

```
for index = j:k
    statements;
end

for index = j:m:k
    statements;
end
```

- j:k is a vector with elements j, j+1, j+2, …, k
- j:m:k is a vector with elements j, j+m, j+2m, ..., such that the final element does not exceed k
- index must be a variable
- Loop statements should be indented

Challenge 1.2

- Write a program that takes in n numbers and displays the average

```
clear;

sum = 0;

n  = input ('How many numbers : ');

for i = 1:n
    num  = input ('Please enter a number : ');
    sum = sum + num;
end

avr = sum/n;

fprintf("The average of the numbers is %f\n",avr);
```

(4) Relational Operators

- Relational operators form a logical expression (condition) that is either true or false
- The basis for decision logic

```
>> 10 > 9

ans =

  logical

   1
```

```
>> 10 == 9

ans =

  logical

   0
```

Relational Operator	Meaning
<	Less than
<=	Less than or equal
==	Equal
~=	Not equal
>	Greater than
>=	Greater than or equal

The one-line if statement

- if condition; statements; end

```
num = input('Enter a number: ');

if num < 0; disp('Negative number'); end
```

Challenge 1.3

- The following statements all assign logical values to the variable x. See can you correctly determine the value of x in each case, before checking the answer in MATLAB

```
x = 3 > 2
x = -4 <= -3
x = 1 < 1
x = 2 ~= 2
x = 3 == 3
```

The if-else construct

if condition

    statementsA

Else

    statementsB

end

```
num = input('Enter a number: ');

if num >= 0
    disp([num2str(num) ' is positive'])
else
    disp([num2str(num) ' is negative'])
end
```

elseif Construct

```
if condition1
    statementsA
elseif condition2
        statementsB
elseif condition3
        statementsC
:::
else
    statementsE
end
```

```
age = input('Please enter your age: ');


if age >= 65
    discount = 0.25;
elseif age < 18
    discount = 0.10;
else
    discount = 0.0;
end

disp(['Age = ' num2str(age) ...
      ' Discount = ' num2str(discount)])
```

Matrix – A Mathematical Definition

- In linear algebra, a matrix is a rectangular grid of numbers arranged into *rows* and *columns*
- An *r x c* matrix has *r* rows and *c* columns
- Here is an example of a 2 x 3 matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

http://samples.jbpub.com/9781556229114/chapter7.pdf

Matrix Name and elements

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

- A is a 3 by 3 matrix
- Uppercase letters usually used for variables that are matrices
- a_{ij} denotes the element in A at row *i* and column *j*

Square Matrices

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

- Matrices with the same number of rows and columns are called *square matrices*
- The diagonal elements of a square matrix are those elements where the row and column index are the same (a_{11}, a_{22}, a_{33})

Diagonal Matrix

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 10 \end{pmatrix}$$

- If all non-diagonal elements in a matrix are zero, then the matrix is a diagonal matrix.

Identity Matrix

$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- A special diagonal matrix is the *identity matrix.*
- This is an *n x n* matrix with ones on the diagonal and zeros elsewhere
- It's special because its the multiplicative identify element for matrices

Vectors as Matrices

- Matrices may have any positive number of rows and columns, including one
- Matrices with one row or one columns are vectors
- A vector of dimension n can be viewed as either:
  - *1 x n* matrix (row vector)
  - *n x 1* matrix (column vector)

Transposition

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}^T = \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & i \end{pmatrix}$$

- Consider a matrix M with dimensions *r x c*
- The transpose of M (M^T) is the *c x r* matrix where the columns are formed from the rows of M.
- $M^T_{ij} = M_{ji}$
- For vectors, transposition turns row vectors into column vectors and vice-versa

Multiplying a Matrix with a Scalar

$$kA = k\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} ka_{11} & ka_{12} & ka_{13} \\ ka_{21} & ka_{22} & ka_{23} \\ ka_{31} & ka_{32} & ka_{33} \end{pmatrix}$$

- A matrix A may be multiplied with a scalar k, resulting in a matrix of the same dimension as A
- The multiplication takes place in a straightforward fashion, each element in the new matrix is the product of k with the corresponding element

Multiplying Two Matrices

$$\begin{pmatrix} ? & ? \\ ? & ? \\ ? & ? \end{pmatrix} \quad \begin{pmatrix} ? & ? & ? \\ ? & ? & ? \end{pmatrix} = \begin{pmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix}$$

3 x 2 2 x 3 3 x 3

- An *r x n* matrix (A) can be multiplied by a *n x c* matrix (B)
- The result (AB) is a *r x c* matrix

Example from MATLAB

A

1 2
3 4
5 6

B

10 20 30
40 50 60

C = A * B

90 120 150
190 260 330
290 400 510

- Let matrix C be the r x c product AB of the *r x n* matrix A with the *n x c* matrix B.
- Each element c_{ij} is equal to the vector dot product of row *i* of A with column *j* of B

2x2 Example

$$AB = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$AB = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

Other information

- Matrix multiplication is not commutative

  $AB \neq BA$

- Matrix multiplication is associative

  $(AB)C = A(BC)$

- Matrix multiplication also associates with multiplication by a scalar or vector

  $(kA)B = k(AB) = A(kB)$

  $(vA)B = v(AB)$

Multiplying a Vector and a Matrix

$$\begin{pmatrix} x & y & z \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} xa_{11}ya_{21}za_{31} & +xa_{12}ya_{22}az_{32} & +xa_{13}ya_{23}ya_{33} \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} xa_{11} + ya_{12} + za_{13} \\ xa_{21} + ya_{22} + az_{23} \\ xa_{31} + ya_{32} + za_{33} \end{pmatrix}$$

- We can multiply a vector and a matrix using the general rules discussed.
- However, the choice of a row or column vector is important

Matrices in MATLAB

- In MATLAB, a matrix is a rectangular object consisting of rows and columns
- A vector is a special type of Matrix (having only on row or one column)
- MATLAB handles vectors and matrices in the same way

Initialising Vectors

- Rules:
  - Elements in the list must be enclosed in square brackets
  - Elements in the list must be separated by either spaces or by commas

```
>> x = [1 2 3 4 5]

x =

     1     2     3     4     5

>>
>> size(x)

ans =

     1     5
```

Sample operations on vectors

```
>> x

x =

   1    2    3    4    5

>>
>> x(1)

ans =

   1

>> x(1:3)

ans =

   1    2    3
```

```
>>
>> sum(x)

ans =

   15

>> mean(x)

ans =

    3
```

Challenge 1.4

- Can you work out what c will be?

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

Initialising Vectors: The colon operator

```
>> x = 1:10

x =

    1    2    3    4    5    6    7    8    9   10

>> x = 1:.5:10

x =

  Columns 1 through 15

    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000    4.5000    5.0000    5.5000
6.0000    6.5000    7.0000    7.5000    8.0000

  Columns 16 through 19

    8.5000    9.0000    9.5000   10.0000
```

Transposing Vectors

- Vectors generated so far are row vectors
- Column vectors may be needed for matrix operations
- The single quote can be used to transpose a vector

```
>> v

v =

     1     2

>>
>> v'

ans =

     1
     2
```

Matrices

- May be thought of as a table consisting of rows and columns
- Has comprehensive support with functions and operators

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

a =

     1     2     3
     4     5     6

>>
>> a'

ans =

     1     4
     2     5
     3     6
```

Subscripts to access matrix elements

- Individual elements are usually referenced with two subscripts (row#, column#)

```
clear;

A = [1 2 3; 4 5 6; 7 8 9];

A(1,1)

A(1,2)

A(2,1)

A(2);
```

Linear Indexing of Arrays

You can refer to the elements of a MATLAB® matrix with a single subscript, A(k). MATLAB stores matrices and arrays not in the shape that they appear when displayed in the MATLAB Command Window, but as a single column of elements. This single column is composed of all of the columns from the matrix, each appended to the last.

So, matrix A

```
A = [2 6 9; 4 2 8; 3 5 1]
A =
     2     6     9
     4     2     8
     3     5     1
```

is actually stored in memory as the sequence

2, 4, 3, 6, 2, 5, 9, 8, 1

A(4) equals 6

https://uk.mathworks.com/help/matlab/math/matrix-indexing.html

Using : with subscripts

- Using the colon (:) operator in place of a subscript denotes all the elements in the corresponding row or column.
- A(3,:) means all the elements in the third row
- A(:,3) means all the elements in the third column

```
clear;

A = [1 2 3; 4 5 6; 7 8 9];

% extract the full row 3
A(3,:)

% extract the column 3
A(:,3)

% extract a subset of the matrix
A(1:2, 2:3)
```

Matrix Functions

Function	Description
eye	Identify matrix
linspace	Vector with linearly spaced elements
ones	Matrix of ones
rand	Uniformly distributed random numbers and arrays
randn	Normally distributed random numbers and arrays
zeros	Matrix of zeros
det	Determinant
eig	Eigenvalues and eigenvectors
expm	Matrix exponential
inv	Matrix inverse
trace	Sum of diagonal elements
{}\ and /	Linear equation solution

Challenge 1.5

- Simulate the roll of 2 dice 10 times using the randi function (an n x 2 array). Call rng(100) before to ensure replication of results.
- Sum both of the outcomes for each throw, using the colon (:) operator to extract all rows and 1 column from a matrix.
- Find the number of 7s in the outcomes. Use vectorisation to store the comparison in a new vector (e.g. Try sum == 7 and check the outcome)
- Check the proportion of these as compared to the expected outcome (statistics theory).
- Try the experiment for 10, 100, 1000 and 10000 rolls

Challenge 1.6

- Write a program that accepts a stream of numbers (-1 to terminate the stream)
- Calculate the minimum, maximum and average
- Make use of the *while* statement