5ELEN018W - Robotic Principles
Lecture 1: Introduction to the Module and Matlab

Dr Dimitris C. Dracopoulos

Introduction to the Module

- Syllabus
- Lectures
- Tutorials (Practicals)
- Software
- Assessment
- Schedule
- What is expected from you?
  - → Lecture Attendance
  - → Tutorial Attendance (actual not just swiping of card!)
  - → Completion of ALL Tutorial Exercises within the week (if not possible within the tutorial session then on your own time).
  - → Raise Issues Early **directly with the Module Leader!**
  - → Code of Conduct

Code of Conduct

- Do not cheat on assignments (this is *INDIVIDUAL* work and **NOT** the product of collaboration!):
  - → Discuss only general approaches not specific details of implementation
  - → Do not take written notes on other's work and do not exchange code
- Cheating is reported to university and then it is out of the module lecturers hands (independent committee decision without the participation of the module tutors)
- Possible consequences:
  - → A mark of 0 for assignment
  - → A mark of 0 for the course
  - → A permanent note on student record
  - → Suspension/Expulsion from university

Code of Conduct (cont'ed)

- Any code found in the web or textbook and used in your work should be properly referenced in comments within your code.

Academic Integrity

- The University of Westminster is committed to the highest standards of academic integrity and honesty. Students are expected to be familiar with these standards regarding academic honesty and to uphold the policies of the University in this respect. Students are particularly urged to familiarise themselves with the provisions of the Academic Regulations and in this case with Academic Misconduct Regulations (https://www.westminster.ac.uk/sites/default/public-files/general-documents/Section-10-Academic-Misconduct-v2.pdf) and avoid any behaviour which could potentially result in suspicions of cheating, plagiarism, misrepresentation of facts and/or participation in an offence. Academic dishonesty is a serious offence and can result in suspension or expulsion from the University.

Topics

- Configuration Space
- Spatial Descriptions and Transformations
- Manipulator Kinematics
- Control
- Inverse Kinematics
- Velocity Kinematics - Jacobian
- Trajectory Generation
- Motion Planning

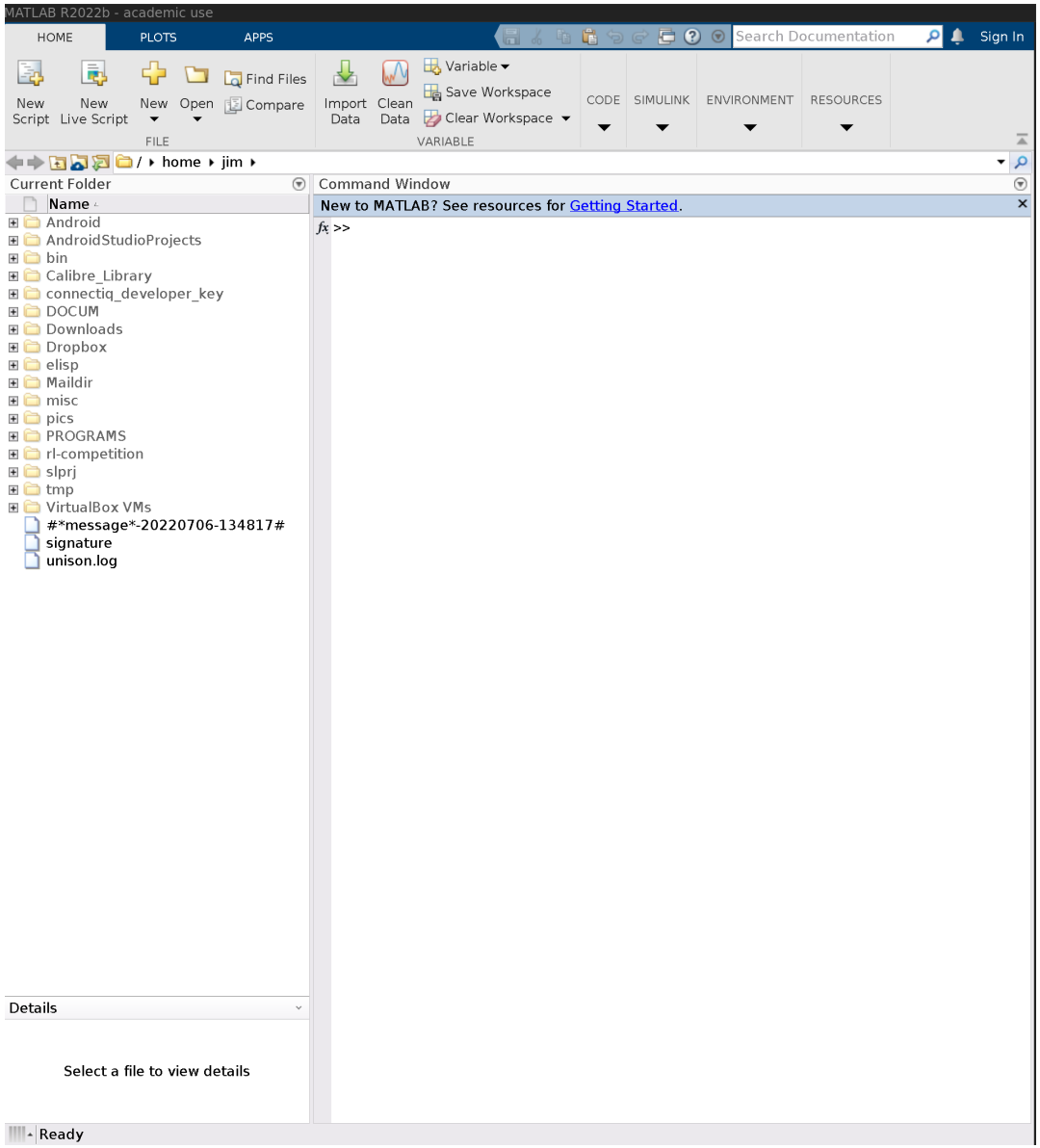
A very mathematical object but will try to simplify! Cannot do without maths!

Introduction to Matlab

Matlab (MATrix LABoratory)

- Software for processing matrices and their functions for numerical calculations in science and engineering disciplines.
- Everything based on matrices!
- High level language - a few lines of code to achieve the equivalent of code in lower level languages such as Java, C++, C#.
- Contains a number of specialised topics toolboxes (e.g. control, robotics). Simulink (block diagram environment for developing systems without writing code)

Matlab's Environment



Help documentation in Matlab

- `help` *topic* - documentation on topic where topic is category or command or function or symbol.
- `lookfor` *keyword* - searches for the specified keyword in the summary line of all the references pages.
- `doc` - help browser for extensive documentation of Matlab and its toolboxes.
- `demo` - demos for Matlab.

Everything is a Matrix

- Matlab manipulates matrices
- In some cases Matlab treats:
  - → matrices 1x1 as scalars
  - → matrices with 1 row or 1 column as vectors.
- Function `size` returns the dimensions of a matrix:
  - → `[M, N] = size(A)`
  - → `size(A,1)`: number of rows of A
  - → `size(A,2)`: number of columns of A
  - → For vectors the function `length` can be used

Statements - Expressions - Variables

- Statements have the form:

  ```
  variables = expression
  ```

  or

  ```
  expression
  ```

  in the second case the expression's value is assigned to special variable `ans`.
- Statements which terminate with a semicolon ; do not produce output on the screen.

Creation of Matrices

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

  or

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

  Elements in the same row can be separated by a space or comma.

- `B = rand(1,5)`

  generates a random matrix of 1x5 (vector) with uniform distribution in the range $(0, 1)$.

Elements in a matrix can be accessed, e.g. $A(1, 2), b(4), b(A(2, 2)$.
Indices start from 1!

Examples of Functions for the creation of Matrices

- `eye(5)` 5x5 Identity matrix (diagonal with 1, the rest of the elements are 0).
- `zeros(5)` 5x5 matrix with zeros.
- `ones(5)` 5x5 matrix with 1.
- `diag(A)`: the diagonal of matrix A
- `magic(N)`: magic square: a matrix with size NxN with integers 1 to N^2 having equal row, column and diagonal sums.

The Colon : Operator

- `i:j` is the vector with elements from i to j with step 1.
- `i:s:j` is the vector with elements from i to j with step s.
- `linspace(a, b, N)` creates a row vector with N equally spaced points between a and b.

Selecting Parts of a Matrix

Select the rows and columns that we are interested in:

```
>> A = [2 3 5; 7 11 13; 17 19 23]

A =

     2     3     5
     7    11    13
    17    19    23

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

ans =

    11    13
    19    23
```

Selecting Parts of a Matrix (cont'd)

Select the rows and columns that we are interested in:

```
>> A(2:3, 1:3)
ans =
     7    11    13
    17    19    23

>> A(:,1)  % first column
ans =
     2
     7
    17

>> A(2,:)  % second row
ans =
     7    11    13
```

Selecting Parts of a Matrix (cont'd)

```
>> A = [2 3 5; 7 11 13; 17 19 23]

A =

     2     3     5
     7    11    13
    17    19    23

>> B = A([1 3], [2 3])

B =

     3     5
    19    23
```

Selecting Parts of a Matrix (cont'd)

Select the rows and columns that we are interested in:

```
>> A(end, end:-1:1)

ans =

    23    19    17

>> B(:)

ans =

     3
    19
     5
    23
```

Operations with Matrices

+	Addition
-	Subtraction
*	Multiplication
^	Power 1
'	Transpose
\	*Left* matrix divide
/	*Right* matrix divide

If matrix A is invertible:

- `X = A\B` is the solution of the system $A * X = B$
- `X = A/B` is the solution of the system $X * B = A$

Use `.* .^ ./ .\` to execute the operations element by element between 2 arrays.

The for loop

```
x = [];
for i = 1:5
    x = [x, i^2];
end
```

The while loop

Example:

```
a = 1
b = 10
while a < b
    a = a + 1
end
disp(a)
```

The final value of a is 10.

The if statement

Syntax:

```
if expression
  statements
ELSEIF expression
  statements
ELSE
  statements
END
```

Scripts and Functions (`.m`-files)

You can create a file with the extension `.m`.

- The file can contain a sequence of Matlab statements which can be executed by calling the file name (without the extension) on the Matlab prompt when you are in the same directory which the file was created.
- The file can contain the definition of a single user-defined function.

Script Example

Create file `myscript.m` with the contents:

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

B = [1 2;
     3 4;
     5 6]

A*B
```

Navigate to the directory which contains your file and type `myscript` in the Matlab prompt to execute it.

Function Example

```
function a = sq(X)
    % Returns a matrix containing the triple of the
    % squared elements of matrix X
    a = 3.*(X.^2);
end
```

Executing the function (with A being the matrix in the previous example):

```
sq(A)
```

outputs:

```
>> sq(A)
ans =
     3    12    27
    48    75   108
```

Creating faster Matlab Programs

- Try to avoid element by element manipulation using loops. Use matrices operations instead.
- Use built-in functions.