ENGG1003 - Tuesday Week 9

Introduction to MATLAB
Variables & Arithmetic
Vectorisation

Brenton Schulz

University of Newcastle

May 6, 2019

Assumed Knowledge

- These notes are written for ENGG1003 and assume C was taught first
- In particular, they require knowledge of:
  - Program top-to-bottom sequential execution
  - Flow control (IF / WHILE / FOR / etc)
  - Variables

What is MATLAB?

- MATLAB is an *interpreted* programming language designed for quickly performing numerical analysis
- It is sometimes criticised for not being a "legitimate" programming language.
  - Engineers use it to solve a complex numerical problem quickly, then throw the code away
  - NB: The code is *written* quickly, it doesn't necessarily *execute* quickly (compared to a compiled language like C)
- Arithmetic is fast, flow control is *very* slow
  - Problems need *vectorisation*

...Interpreted?

- A language is *compiled* when the entire source code listing gets converted to a binary executable in one step
- An *interpreted* language is read and executed line-by-line
    - The language interpreter is running the whole time your code is running

...Interpreted?

- A language is *compiled* when the entire source code listing gets converted to a binary executable in one step
- An *interpreted* language is read and executed line-by-line
  - The language interpreter is running the whole time your code is running

...Interpreted?

- Interpreted languages are slower, but have the advantages of:
  - Being more forgiving of mistakes
  - Having more advanced memory management, eg:
    - Variables don't need to be declared
    - Arrays automatically grow and shrink as needed
  - Allowing code snippets to easily be executed in isolation

MATLAB Vs C

- Some big contrasts:
  - MATLAB is "weakly typed"
    - There are no strict data types
    - By default, (almost) everything is a complex valued array of type `double`
  - Arithmetic (mostly) follows rules of linear algebra
    - Somewhat beyond this course. We won't cover matrix multiplication.
    - Many language behaviours will make more sense after you've studied linear algebra
    - The fact that "everything is an array" makes for some possibly confusing rules
  - MATLAB has "high level" features like plotting
    - It is more of a "calculator engine" than a programming language

Installing MATLAB

- MATLAB is (expensive) commercial software
  - Python is more popular in industry (c.f. IEEE survey), partly because it is free
- The university pays a site licence which allows students to install it for free
  - Instructions here (hopefully...): `https://uonau.service-now.com/itservices?id=kb_article_view&sysparm_article=KB0023081&sys_kb_id=a7ccc3334f3953c08e8fa90f0310c7f7`
- The "standard" licence (for companies) is $1260 per year, per computer

Installing Octave

- Octave is a cost-free (and open source) MATLAB-like interpreter
    - Some employers prefer this over MATLAB
- It will probably execute all of the code for this course without modification
- Available for Windows / Mac / Linux: `https://www.gnu.org/software/octave/download.html`
- Demonstration of projects in Octave is fine
    - It tends to load *much* faster than MATLAB

Variable Classification

- MATLAB may be "weakly typed" but the following classifications are useful:
  - A *scalar* is a single number
  - A *vector* is a row or column of numbers
    - A 1D array in C
  - A *matrix* is rectangular array of numbers
    - A 2D array in C

Variable Classification

- MATLAB may be “weakly typed” but the following classifications are useful:
  - A *scalar* is a single number
  - A *vector* is a row or column of numbers
    - A 1D array in C
  - A *matrix* is rectangular array of numbers
    - A 2D array in C
- Arithmetic operations have different behaviours with different arguments, especially when mixed (eg: what does scalar plus vector do?)

Getting Started

- Lets load up MATLAB and:
  - Learn what the different GUI segments do
  - Allocate values to some random variables
    - Observe them appear in the "workspace"
  - Do some basic arithmetic on scalar variables
  - Run a basic script
  - Observe output suppression

Variable Allocation Syntax

- When allocating a constant to a variable we have a few basic methods:
  - Scalar: just like in C

    ```
    x = 5
    ```

  - Row Vector: space separated list inside `[ ]`'s

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

  - Column Vector: like row vectors, but uses `;` to separate rows:

    ```
    x = [1;2;3;4]
    ```

  - Matrix: A mix of row and column syntax:

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

Observations

- First, run the code from the previous slides

Observations

- First, run the code from the previous slides
- Observe that the variables are created automatically

Observations

- First, run the code from the previous slides
- Observe that the variables are created automatically
- Observe that MATLAB prints the result after each line
- Add a `;` to the end of a line to suppress the output

Observations

- First, run the code from the previous slides
- Observe that the variables are created automatically
- Observe that MATLAB prints the result after each line
- Add a `;` to the end of a line to suppress the output

Arithmetic

- For scalar data, MATLAB supports all basic arithmetic operators just like C
- It also supports exponents with ^
  - Shift-6 on a US keyboard
  - In C, this means a bitwise exclusive-OR