ENGG1003 - Tuesday Week 9

Introduction to MATLAB
Variables & Arithmetic
Vectorisation

Brenton Schulz

University of Newcastle

May 7, 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?

- ▶ 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
        - ▶ You can run single lines instantly by typing them into a command prompt

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

Other Initialisation Methods

▶ MATLAB supports many other methods for "creating" data:

  ▶ Create a list of numbers from A to B with increment C with:

```
x = A:C:B
```

  eg:

```
x = 0:0.1:2
```

  ▶ Use the `linspace()` function
    ▶ View the help page and run an example
  ▶ We can also read data from files; see this later

Arithmetic

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

Arithmetic

- ▶ For vector and matrix data addition and subtraction is *element-wise*
  - ▶ The arguments should be the same size
    - ▶ Or *at least one* dimension must match but this is beyond ENGG1003 (and only supported on recent versions of MATLAB)
  - ▶ If the argument dimensions differ an error occurs
- ▶ Run some examples...

Arithmetic

- ▶ Vector and matrix multiplication and division are beyond ENGG1003
  - ▶ You will learn matrix multiplication in MATH..something
  - ▶ Division is advanced: multiplication by *matrix inverse*

Arithmetic

▶ Vector and matrix multiplication and division are beyond ENGG1003
  ▶ You will learn matrix multiplication in MATH..something
  ▶ Division is advanced: multiplication by *matrix inverse*
▶ However, you can do element-wise multiplication if the arguments are the same size
  ▶ This is done with special operators:
    ▶ .* Multiplication
    ▶ ./ Division
    ▶ .^ Exponent
▶ Run some examples...

Arithmetic

- ▶ Final examples (for now):
    - ▶ Scalar plus/minus vector/matrix adds/subtracts that scalar from every element
    - ▶ Scalar multiplication does the same
    - ▶ Scalar divided by vector/matrix causes an error (unless `./` is used)
    - ▶ Vector/matrix divided by scalar does an element-wise operation
- ▶ Run some examples...

Maths Functions

▶ MATLAB has hundreds (thousands?) of built-in functions

▶ Lets play with trig functions

▶ Run an example of *scalar* `sin()`

Maths Functions

▶ MATLAB has hundreds (thousands?) of built-in functions
▶ Lets play with trig functions
▶ Run an example of *scalar* `sin()`
▶ What about vectors and matrices?

Maths Functions

▶ MATLAB has hundreds (thousands?) of built-in functions
▶ Lets play with trig functions
▶ Run an example of *scalar* sin()
▶ What about vectors and matrices?
▶ Mathematics doesn't define what a "matrix sine" (etc) is
▶ MATLAB simply evaluates them element-wise
▶ Run an example

Scripts

▶ MATLAB supports running source code from a file

▶ There is a built-in editor we can use to write (and debug) said files

▶ Long story short:
  ▶ Create a file with the editor
  ▶ Save it with the .m extension
  ▶ Click the "run" button

▶ Script files can also be run from the command prompt by typing their name *without the .m*
  ▶ Prompt must have the correct "working directory"

Plotting

- ▶ MATLAB has very powerful built-in plotting tools
    - ▶ They are commonly used in academic literature
    - ▶ They make Excel plots look like something my 4 year old drew at daycare
    - ▶ An impressive example I found on Google images:
      `http://www.asu.cas.cz/~bezdek/`
      `vyzkum/rotating_3d_globe/rotating_`
      `3d_globe/manual.html`

Plotting

- ▶ Basic 2D plotting is done with the `plot()` function
- ▶ It does the Excel equivalent of a scatter plot
- ▶ The first two arguments are vectors (1D arrays) listing x-y pairs of points
- ▶ Run a basic example...

Plotting

▶ We can finally bring a few things together to do an interesting plot

Plotting

- ▶ We can finally bring a few things together to do an interesting plot
- ▶ Task: plot `sin(x)` from 0 to 2π
- ▶ Steps
    1. Create a vector, `x`, that contains "many" elements going from 0 to 2π
    2. Pass this vector to the `sin()` function
    3. Run the plot command as plot(x, sin(x))
- ▶ Do the above manually then repeat it from a script

Scripts And Scalar Arithmetic Example

▶ Example: (From last year's slides) Write a
MATLAB script which, given adequate physical
data, calculates the rate at which the Sun loses
mass due to nuclear fusion