Computer Algebra and Technical Computing (MTH1006)

B. Vorselaars
bvorselaars@lincoln.ac.uk

School of Mathematics and Physics, University of Lincoln

Module structure

- Contents:
  - Matlab (Technical Computing): 1st term
  - Maple (Computer Algebra): 2nd term
- 15 credits spread over the whole academic year. 12 weeks per part. Learning load on average 6 hours 15 minutes per week (all-inclusive).
- Grading
  - 20% logbook (one logbook for Matlab and one for Maple)
  - 80% coursework (one out-of-class assignment, one in-class test for Matlab; same for Maple).

Assessment dates - Matlab

- Assessment dates first term
  - 24-10-2023: 1st coursework handed-out
  - 10-11-2023: hand-in 1st coursework
  - 8-12-2023: hand-in logbook Matlab
  - 12-12-2023: Final in-class test Matlab
- See also blackboard, heading 'Assessments'

Typical weekly structure

A session consists of two hours, typically split into three parts

- First part: discussion new material, setting exercises.
- Second part: you can try to solve the exercises yourself and there is help available
- Third part: discussion solution exercises of previous session (if applicable), help if you are stuck

Logbook

Why a logbook for this module?

- All the information entered on the computer will otherwise be lost and you would have no trace of your work (as what you would have with hand-written notes).
- More generally, another person should understand and can reproduce what you did by reading it. They are used in the process of scientific research.

Logbook

What is a logbook?

Logbook

Aka notebook. A diary of your work activities. Should be self-explanatory and are an element of scientific research.

A log book is a collection of entries, with each entry containing

- your name and date
- assignment/problem/exercise
- (partial) solutions, including pseudo-code and/or flowcharts (see `draw.io`)
- summary of what you have learned
- reflective self-assessment of learning process: what did you understand, what was more challenging?
- references (if appropriate)

Logbook

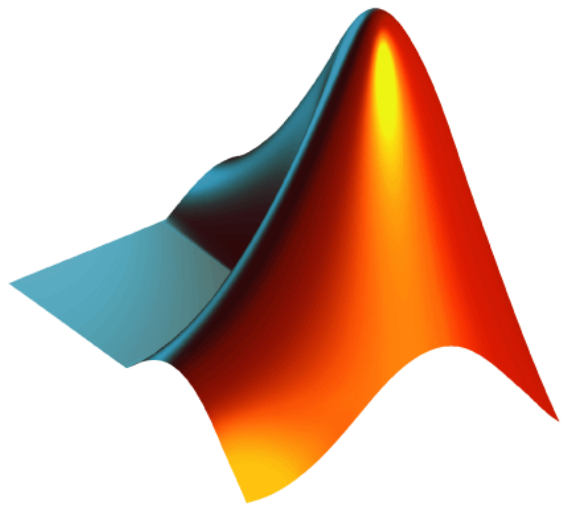
How to construct a logbook? Most convenient: electronic logbook (as opposed to hand-written).demo

- Copy-paste figures and text to Word. A good option is to use the online version of Word, by going to `https://onedrive.lincoln.ac.uk` and create a new Word document. Then the document is automatically saved. Don't forget to logout at the end of the session.
- Text can be copy-pasted by selecting the text, then pressing *CTRL+C* in Matlab, and press *CTRL+V* in Word.
- (Windows 10) Use *WINDOWSKEY+SHIFT+S* buttons to take a screen shot in Matlab. Paste it with CTRL+V in a word document to store it.

demoMore information: description and example on blackboard.

Matlab Logbook (example)

COMPUTER ALGEBRA AND TECHNICAL COMPUTING



Bart Vorselaars | MTH1006 | 2023-2024
University of Lincoln, School of Mathematics and Physics

Declaration

I confirm that this logbook is all my own work and that all references and quotations from both primary and secondary sources have been fully identified and properly acknowledged – Bart Vorselaars

Front cover: *picture taken from* http://uk.mathworks.com/company/newsletters/articles/the-mathworks-logo-is-an-eigenfunction-of-the-wave-equation.html

Table of Contents

Week 7

....

Week 8

15-11-2023
In week 8 we learned about the importance of efficient programming. In MATLAB timing a code can be carried out using the statements *tic* and *toc*.

One of the computationally intensive operations is to grow an array in a loop. Much quicker is to create the array before the loop and then fill in the values afterwards.

EXERCISES

Timing

1

A

The script is (copy-pasted from MATLAB to Word)

```
% The following script calculates the double sum:
%  S=sum_{n=1}^N sum_{m=1}^N delta_{nm}*n
% for N=10000. Here delta_{nm} is the Kronecker
% delta function, which is 1 for n=m and zero otherwise.
N=10000;
S=0;
for n=1:N
    for m=1:N
        if n==m
```

```
            d=1;
        else
            d=0;
        end
        S=S+d*n;
        disp(['n,m,d,S=',num2str([n,m,d,S])])
    end
end
S
```

The delta function is coded using the variable *d*. I've included a display statement (for debugging purposes) and it can be disabled by putting a percentage sign (%) in front of it. Each iteration within the two *for* loops the variable S is increased, so that at the end of the two *for* loops it should contain the requested quantity.

B

The script can be rewritten as a function with name *S1.m*:

```
function S=S1(N)
% S=S1(N)
%
% Returns the double sum:
%  S=sum_{n=1}^N sum_{m=1}^N delta_{nm}*n
% Here delta_{nm} is the Kronecker delta function,
% which is 1 for n=m and zero otherwise.
S=0;
for n=1:N
    for m=1:N
        if n==m
            d=1;
        else
            d=0;
        end
        S=S+d*n;
        %disp(['n,m,d,S=',num2str([n,m,d,S])])
    end
```

```
end
```

Here the *disp* function call has now been disabled, otherwise too much output is generated for large *N*. To call this function, the following script has been used:

```
% Test script for testing the function S1 with a known value.
N=1;
S=S1(N);
assert(S==1,'Unexpected value of the sum: for N=1 the sum S should be 1')
```

C

The function can be simplified using one for loop, with the function *S2.m*:

```
function S=S2(N)
% S=S2(N)
%
% Returns the sum
%  S=sum_{n=1}^N n
S=0;
for n=1:N
    S=S+n;
end
```

D

The function can be simplified further, since there is an analytical expression for the sum of elements ranging from 1 to N. This results in the function *S3.m*:

```
function S=S3(N)
% S=S3(N)
%
% Returns the sum
%  S=sum_{n=1}^N n
% which can be simplified to the analytical expression
%  S=0.5*(N^2+N)
S=0.5*(N^2+N);
```

E

We can call all three functions subsequently using the following script:

```
% Timing script for the three different functions.
% Each of them calculate the same quantity, but with
% a different computational complexity.
N=10000;
tic
S=S1(N);
toc
tic
S=S2(N);
toc
tic
S=S3(N);
toc
```

Running it gives the timing results for N=10000:

```
Elapsed time is 1.481639 seconds.
Elapsed time is 0.000130 seconds.
Elapsed time is 0.000024 seconds.
```

This shows that the S_3 function is fastest. However, this only occurs when taking the minimum values after repeated execution of the script. Otherwise, because of other program running the timing could be very different:

```
Elapsed time is 6.185825 seconds.
Elapsed time is 0.004680 seconds.
Elapsed time is 0.013501 seconds.
```

Doubling N to 20000 results in:

```
Elapsed time is 6.391665 seconds.
Elapsed time is 0.000241 seconds.
Elapsed time is 0.000025 seconds.
```

This shows that function S1 is approximately 4 times slower, S2 approximately 2 times, and S3 approximately constant. For N=40000 one would expect that S1 runs by approximately 16 times slower, S2 approximately 4 times, and S3 would be independent on N. The reason is that in the function S1 the addition to the sum gets executed N*N=N^2 times, so that it needs N^2 operations. For S2 the addition to the sum gets executed N times, so it needs N operations. For S3 there is only 1 line to execute and it therefore only needs 1 operation. The time it takes for each operation is roughly constant, so that upon doubling N: the time for S1 quadruples, the time for S2 doubles and the time for S3 stays constant.

GENERAL REFLECTION ON MATERIAL

This week was not too difficult to understand and master, apart from implementing the double 'for' loop. However, after consulting more examples in the book by Stormy Attaway I managed that as well.

References

While solving some of the exercises, background information on the various commands is taken from:

- The MATLAB documentation within the program itself, version R2023a, using *help* and *doc*
- Stormy Attaway, *Matlab, a practical introduction to programing and problem solving*, third edition, Elsevier, Amsterdam, 2013

Marking scheme logbook

Item	Max points
Title	5
Student name	5
Dated entries	5
Page numbering	5
Declaration own work	5
Table of contents	5
References	5
Structure (sections and subsections titles, appropriate font (bold if necessary, readable size), appropriate size of images, standard margins)	5
Complete record of weekly work in class	20
Complete record of self-study outside class hours	10
Clear and complete records including detailed reflection on learning process, such as own achievements and difficulties, what has been learned and reference to module LO's	30
Total	**100**

Learning objectives of Technical Computing with Matlab

- Implement simple numerical algorithms using technical computing.
- Maintain a well structured computational laboratory log-book.
- How? Practise, practise, practise

Matlab will be used in other modules and in projects

Main contents technical computing

- Control structures of a procedural language including for-loops and while-loops; if, if-else, nested-if and switch statements
- Designing solutions of coding problems
- Matlab language syntax, data input & output, graphical display of data, creating Matlab functions and Matlab script files.
- Variables and data types including integers, floating point numbers, strings, 1D and 2D arrays and how to address and modify the elements in these arrays
- Writing and debugging computer programs in Matlab

Programming

Any famous programmers?

Programming

Figure: Ada Lovelace (first computer programmer)

Programming

Figure: Bill Gates (Microsoft)

Programming

Figure: Mark Zuckerberg (Facebook)

Programming

Why for Mathematics and Physics degree?

- Counting till 10: use your fingers!
- Short algebraic manipulations: paper and pen
- Simple standard functions on real numbers: pocket calculator
- Planetary motions: computer programming
- Financial mathematics: computer programming
- Complex algebraic manipulations: computer programming

Endless more examples: large prime numbers, cryptography, proof of complex mathematical theorems (the four color theorem), nuclear physics (CERN, first usage of internet), weather predictions, material modelling, etc.
(`http://uk.mathworks.com/company/user_stories`)

Accessing Matlab

A license is needed, but the university has a license to allow each student to install it at home on their own computer, and it is accessible in the Library. demo

- Matlab can be downloaded from `https://uk.mathworks.com/downloads/`. Register with your university email address. See blackboard for detailed installation instructions. This is the advised method.
- Matlab can also be run interactively online from `https://matlab.mathworks.com/`. Some things may be a bit more complex, such as file access. Details are on blackboard.

Alternatives to Matlab

There are free alternatives:

- Octave: `www.octave.org`, to a large degree compatible with Matlab. See also `https://www.tutorialspoint.com/execute_matlab_online.php` for an online version of *Octave*.
- Ipython notebook: `ipython.org`, less compatible, but overlapping functionality.

User interface

Matlab consists of several parts:

- Middle of the screen: command window with prompt >>.
- Top middle: current folder
- Top left: contents of current folder

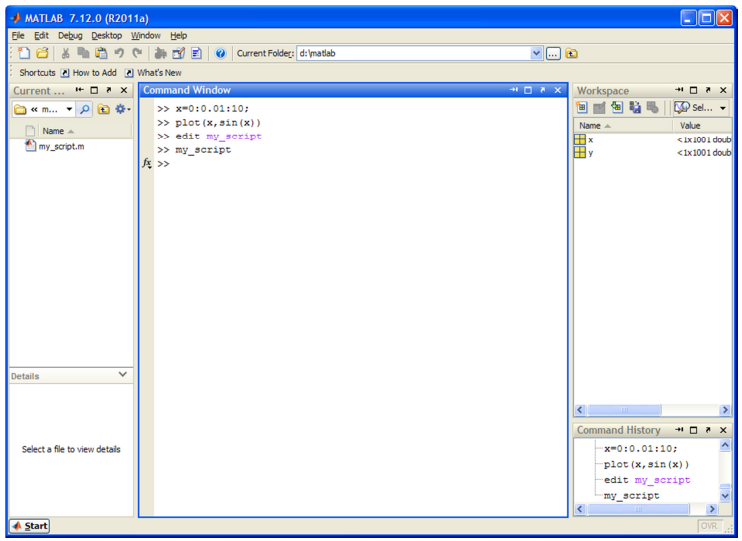


Figure: The user interface of Matlab

demo

Calculator I

Matlab: in its simplest form is a calculator

Addition

```
>> 1+1
ans =
     2
```

Subtraction

```
>> 1-23
ans =
   -22
```

Multiplication

```
>> 2*3
ans =
     6
```

Division

```
>> 3/4
ans =
    0.7500
```

Exponentiation

```
>> 10^2
ans =
   100
```

demo

Calculator II

Matlab: in its simplest form is a calculator

$\sqrt{2} \approx 1.4142$

```
>> sqrt(2)
ans =
    1.4142
```

$e^3 \approx 20.0855$

```
>> exp(3)
ans =
   20.0855
```

$\sin(\pi/2) = 1$

```
>> sin(pi/2)
ans =
     1
```

$\ln(100) \approx 4.6052$

```
>> log(100)
ans =
    4.6052
```

$\log(100) = 2$

```
>> log10(100)
ans =
    2
```

demo

Calculator III

Scientific or exponential notation.
Example: 2.3×10^4

```
>> 2.3 * 10^4
ans =
     23000
```

Shorter way using the symbol e:

```
>> 2.3e4
ans =
     23000
```

Similarly: 1.24×10^{-3}

```
>> 1.24e-3
ans =
     0.0012
```

Help I

Command line help. For basic help on a command, such as `log`, type

```
>> help log
```

```
log    Natural logarithm.
    log(X) is the natural logarithm of the elements of X.
    Complex results are produced if X is not positive.

    See also log1p, log2, log10, exp, logm, reallog.

    Other functions named log

    Reference page in Help browser
       doc log
```

demo

Help II

Command line help. For extended help on a command, such as `log`, type

```
>> doc log
```

demo

Operator precedence

Matlab uses normal operator precedence:

1st exponentiation

2nd unary plus/minus

3rd multiplication/division

4th addition/subtraction

Use parenthesis if needed

```
>> -1*-1
ans =
     1
>> -1^2
ans =
    -1
>> (-1)^2
ans =
     1
```

Special numbers I

- pi: π

```
>> pi
ans =
    3.1416
```

- Infinity: ∞

```
>> inf
ans =
    Inf
>> 1/0
ans =
    Inf
```

Special numbers II

- Not-a-number. Result is undefined

```
>> nan
ans =
     NaN
>> 0*inf
ans =
     NaN
>> 0/0
ans =
     NaN
```

Variables I

By means of variables values can be stored and retrieved. A variable has a *name* and *content*.

- Setting the content of variable `x`: use the *name* and *equal sign*

```
>> x=3
x =
      3
```

- Retrieving content: use just the name

```
>> x
x =
      3
>> 10*x^2
ans =
     90
```

Variables II

The equality sign = differs from the usual mathematical definition.
In Matlab = means be set to

- Retrieving and setting content at the same time

```
>> x=2023
x =
      2023
>> x=x+1
x =
      2024
```

- Deleting variable

```
>> clear x;
>> x
??? Undefined function or variable 'x'.
```

Variables III

The variable of which a value has to be assigned to should always appear *left* of the equality sign.

```
>> x=3
x =
     3
>> 3=x
```

Variable naming

The name of a *variable* in Matlab should obey certain rules:

- Start with a letter
- Potentially continues with a letter, underscore, or digit (multiple times)
- No other characters, such as spaces, %, $, (, +, etc.
- Should not be one of the keywords, such as `end`

✓ `x`
✓ `Y`
✓ `Yes`
✓ `My_long_variable_name_with_1_number`
✓ `x3`
× `3x`: starts with a number
× `x x`: contains a space
× `end`: is a reserved keyword
✓ `sin`: but not advised, since it is also a function
× `x$`: contains the $ sign.

Homework

- Register online for a Matlab account using your university email address
- Complete the entire online Matlab OnRamp course. See `https://matlabacademy.mathworks.com/R2023a/portal.html?course=gettingstarted` When you finish the whole Matlab Onramp course, include a screen dump of your progress in your logbook.
- Install Matlab on your own computer