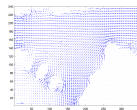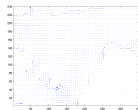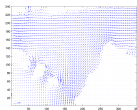# Introductory Lecture
CS319: Scientific Computing (with MATLAB)

Dr Niall Madden

Week 1: 11 January, 2023

Outline
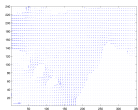
1. **Overview of CS319**
   - Who we are
   - Course materials, lectures, etc
   - Assessment
   - Class times

2. **CS319: what and why**
   - Programming Platform

3. **Scientific Computing**

4. **MATLAB**
   - Motivation: why MATLAB?
   - What is MATLAB?
   - Getting started
   - Fundamental features
   - Output

Lecturer details:

Who: Dr Niall Madden (he/him), School Mathematical and
Statistical Sciences.

Where: Office: AdB-1013, Arás de Brún.

Contact: Email: Niall.Madden@UniversityOfGalway.ie

Students: 3rd year Mathematics and/or Computing (3BS9); 3nd
and 4th year Mathematical Science (3BMS2+4BMS2), 4th Year
Maths/Applied Maths (4BS2); MSc Mathematics; Visiting
Students.

This module will be taught using a mix of in-person "classes", and lab sessions.

- The "classes" will mix conventional lectures, and practical sessions. That is, you will spend some time coding in every class.

- All slides and lecture materials (such as sample programmes) will be made available in advance of the class.

I use Blackboard for

- Posting announcements (1 per week, usually);
- Posting grades
- Assignment uploads
- *Links* to slides and scripts from class.

I'll also use a `git` repository on BitBucket to store files (e.g., these slides, and all MATLAB examples presented in class).

This is because it allows me to push fixes easily. More importantly, you get to honestly state that you've used `git`.

The repository is at:

https://bitbucket.org/niallmadden/2223-cs319

You've been sent an invitation to it. If you didn't get it, send me an email.

In the short-term, slides will also be mirrored at

https://www.niallmadden.ie/2223-CS319/Week01

The notes for CS319 are largely self-contained. But some books will be VERY helpful.

- Scientific Computing with Case Studies (Diane O'Leary)
  https://epubs-siam-org.nuigalway.idm.oclc.org/doi/book/10.1137/9780898717723

- Learning MATLAB (Toby Driscoll).
  https://epubs-siam-org.nuigalway.idm.oclc.org/doi/book/10.1137/1.9780898717662

- MATLAB Guide (Higham and Higham)
  https://epubs-siam-org.nuigalway.idm.oclc.org/doi/book/10.1137/1.9781611974669

- Numerical Computing with MATLAB (Moler)
  https://uk.mathworks.com/moler/index_ncm.html

- Insight through Computing (Charlie Van Loan and Daisy Fan)
  https://epubs-siam-org.nuigalway.idm.oclc.org/doi/book/10.1137/1.9780898717648

- More will be added as needed...

The final grade for CS319 will be based on

- **Four programming assignments** (40%)
- a mid-semester open-book test (20%) (Week 7, approx).
- a project (40%)

This module does not have an end-of-semester exam.

IGNORE THIS SLIDE

	Mon	Tue	Wed	Thu	Fri
9 – 10			✓		
10 – 11					
11 – 12					
12 – 1					
1 – 2					
2 – 3					
3 – 4					
4 – 5			✓		

We need to find some other times for labs.
Please send your time-table to Niall
(Niall.Madden@UniversityOfGalway.ie) ASAP, preferably today.

CS319: what and why

In CS319 we are primarily concerned with two issues:

1. How to use a computer to solve a scientific problem. That is:
   - how to determine the best algorithm to apply in a given situation.
   - how to understand the potentials and limitations of the algorithm.

2. Implementing that algorithm: How to write the code in MATLAB.

More deeply, this is a course on **programming and problem-solving in MATLAB**.

The primary learning outcome is that, by the end of the semester, you will be a reasonably proficient MATLAB programmer, and can honestly list "object-oriented programming in MATLAB" as one of your skills in your CV.

As well as learning how to program in MATLAB, you will learn about some key concepts in Scientific Computing. These include

- Understanding how computers represent numbers;
- Defining functions (of various types);
- Visualisation of functions and data;
- Root-finding and optimisation;
- Data fitting, and least squares;
- Storing dense and sparse arrays (matrices);
- Efficiency and complexity of algorithms (from an experimental/applied view).
- Solving linear systems by direct and iterative methods;
- Representation and visualisation of graphs and networks

University of Galway has a "campus wide licence" for MATLAB, which, actually means you can access it off-campus:

- You can install MATLAB on your own computer. But only do this if you have lots of free disk-space. MATLAB is quite bloated. The install package is 180M, and once installed it takes up nearly 4Gb of space.
- With a network connection, you can use a web-based version at `https://matlab.mathworks.com` (**recommended**)

In either case, you need to create a Mathworks account, linked to your university account. Do this at
`https://uk.mathworks.com/academia/tah-portal/`
`national-university-of-ireland-galway-31528171.html`

Scientific Computing

Dianne O'Leary describes a **computational scientist** as someone whose focus is the intelligent use of mathematical software to analyse mathematical models.

These models arise from problems formulated by scientists and engineering. Solutions/models can then be constructed using statistics and mathematics. Numerical methods are then employed to design algorithms for extracting useful information from the models.

In scientific computing, we are interested in the correct, reliable and efficient implementation of these algorithms. This requires knowledge of how computers work, and particularly how numbers are represented and stored.

History has shown that mistakes can be very, very costly.

Source: Wikipedia

MATLAB

In order to be able to write the code to implement a nontrivial algorithm, one needs to a good grasp of a programming language. To make efficient use of our, and the computer's, time, we will use MATLAB.

1. **Motivation**: what can we do more easily in MATLAB than in other languages.

2. **The basics**: Input and output; data types; arithmetic; loops; Flow of control (`if` statements); conditionals; ploting data; and functions.

3. File input and output.

4. Vectors, matrices, cells.

5. **Abstract data types**: objects and classes.

6. Programming utilities, such as the profiler.

Why MATLAB? Good question!

- We could use Python+NumPy, but many of you have already studied that. And many haven't, which would also be a problem.

- We could use C or C++ (and I used to), but learning all about the compiler and intricacies of the language gets in the way of the Scientific Computing.

- We could use Julia, but maybe not yet (come back in a few years)...

- **More positively**...

- The university has a campus licence for MATLAB.

- It is VERY efficient, allowing us to work on difficult problems.

- It is easy to get started with: it has lots of features, and the syntax is not complex.

- Though barely in the top 20 of popular languages, there are many companies, especially in engineering and finance, that use MATLAB.

- The company that produces MATLAB has its European headquarters in Galway, and is often hiring!

- As mentioned, you should get started at `https://matlab.mathworks.com`. Please do this ASAP.

MATLAB is both a programming language and a software environment.

To start with, we'll use the interactive environment (REPL), later we'll write code as scripts or functions, stored in files.

We'll also use "Matlab Live Scripts", which mixes text and code – a little like Jupyter.

If you prefer to use free software, try **Octave**. Both online and downloadable versions are available. It is mostly compatible with MATLAB.

MATLAB is a standard tool for numerical computing in industry and research. It specialises in matrix computations: it's name is abbreviated from (**Mat**rix **Lab**oratory).

It also includes functions for graphics, numerical integration and differentiation, solving differential equations, image and signal analysis, and much more. The full list of toolboxes includes:

Aerospace, Audio Toolbox, Automated Driving (Roadrunner), Bioinformatics, Communications, Computer Vision, **Curve Fitting**, Data Acquisition, Database, **Deep Learning**, **Econometrics and Financial**, **Optimization**, GPU Coder, **Image Acquisition and Processing**, Mapping and Navigation, **Web App** Server, Parallel Computing, PDEs, 3D Animation, Symbolic Math,...

1. MATLAB is typically used as an interpretative environment. Start MATLAB, and type a single instruction at the `>>` prompt, and it will be run immediately.

2. **Assignment**: to create a variable, you just assign it a value, e.g., `>> t=10`

3. In MATLAB, (almost) everything is a *matrix*. A scalar variable is just a 1×1 matrix. To check this, set
   `>> t=10`
   and use `size(t)` to find the numbers of rows and columns of *t*.

4 To declare a row-vector array, try:

```
>> x = [-4, -3, -2, -1, 0, 1, 2, 3, 4]
```

Or, more simply,

```
>> x = -4:4
```

To access, say, the 3rd entry

```
>> x(3)
```

5 In mathematics, we usually think of vectors as *column* vectors.
To define one, try

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

Or you can take the transpose of a row vector:

```
>> x = [1,2,3]';
```

6 If you put a semicolon at the end of a line of MATLAB, the line is executed, but the output is not shown. (This is useful if you are dealing with large vectors).
If no semicolon is used, the output is shown in the command window.

7 We'll often want to run a collection of commands repeatedly. So, rather than type them individually, create a file with the following code

```
x=-4:4
for i=1:9
  y(i) = cos( x(i) );
end
plot(x,y);
```

Save this as, say `Example01.m`. To execute it, just type `>> Example01` in the MATLAB command window. Or click on "Run" in the editor.

Your file is an example of a MATLAB *script file*.

8 If the picture isn't particularly impressive, then this might be because MATLAB is actually only printing the 9 points that you defined. To make this more clear, use
`plot(x, y, '-o')`
This means to plot the vector y as a function of the vector x, placing a circle at each point, and joining adjacent points with a straight line.

9 The plot generated is not particularly good one. The points plotted are a unit apart. To get a better picture, try `fplot`, as follows `>> fplot(@cos,[-4,4])`

10 The script file from Part (5) is a little redundant. In MATLAB, most functions can take a vector or matrix as an argument. So, in fact, we can just use
`>>  y = cos( x )`
which sets y to be a vector such that $y_i = \cos(x_i)$.

11 MATLAB has most of the standard mathematical functions:
    `>> sin`, `>> cos`, `>> exp`, `>> log`, `>> sqrt`, etc.
    In each case, write the function name followed by the
    argument in round brackets, e.g.,
    `exp(x)`   for e^x.

12 The * operator performs matrix-matrix multiplication. For
    element-by-element multiplication use `.*` For example,
    `>> y = x.*x` will set $y_i = (x_i)^2$.
    So does `y = x.^2`.

Variables are **dynamic** in MATLAB. To create a variable, just assign it a value: `>> a = 23.2`

Later we'll learn about the **scope** of these variables, but that is not needed right now.

To check which variables you have defined use `>> who`. For more detail, such as "class" (data type), and memory usage, try
`>> whos`
For example, after running some of the commands above, when I do this, I get

```
Name        Size              Bytes  Class
a           1x1                   8  double
i           1x1                   8  double
x           1x9                  72  double
y           1x9                  72  double
```

This emphasises that MATLAB's most important data type (class) is a **matrix**. Even a single number is thought of as a 1×1 matrix, usually of type **double**. We'll learn a little more about that next week.

- MATLAB is case-sensitive. E.g., the variables `a` and `A` can store different values.

- Program blocks are terminated by the `end` keyword. That is: unlike Python, white-space is not structural.

- A (logical) line is terminated a "new line"; use "..." for line continuation.

- Use the percent sign `%` to start a comment – everything after them is ignored until an end-of-line is reached.

- A double percent, `%%`, is used to comment a new section in the program (but this is just a feature of the IDE).

Example01.m

```matlab
1  %% CS319 - Example01.m
   % A very simple example of a MATLAB script

   x=-4:4
5  for i=1:9
       y(i) = cos( x(i) );
7  end
   plot(x,y);
```

To finish, because it is traditional, we'll see how to output/display text and variables.

- If you wish to display a variables value, just type its name/identifier in the command window, without a semicolon.
- Or you can use the `disp()` function: `>> disp(x)` This can be useful with a program.
- You can control, a little, how numbers are formatted when displayed using the `format` command. E.g., `>> format long`.
- There are lots of other options. To check, use `>> doc format`

But once we get to progamming properly, we'll use the `fprintf()`
function, which can mix text and different data types, and format
them precisely.

Hello02.m

```
%% CS319 - hello02.m
% The "hello world" script in MATLAB
fprintf('Hello, World!\n');
```