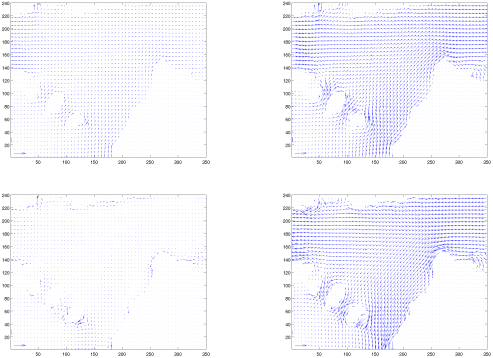
Annotated slides: Wednesday

CS319: Scientific Computing

Introduction to CS319

Dr Niall Madden

Week 1: 15+17 January, 2025



Outline

1 Overview of CS319
- Who we are
- Classes
- Class times
- Materials
- Text books
- Assessment

2 CS319: what and why
- But why C++?
- Best of both worlds?

3 Scientific Computing
- Major topics

4 A first example
- Python
- Octave/MATLAB
- C++

5 Introduction to C++
- Programming Platform

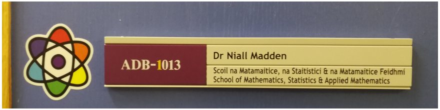
6 Getting started with C++
- Topics
- Programming Platform
- From Python to C++

7 Basic program structure
- “hello world”

Overview of CS319

Who we are

Lecturer details:



ADB-1013

Dr Niall Madden

Scoil na Matamaitice, na Staitisticí & na Matamaitice Feidhmí
School of Mathematics, Statistics & Applied Mathematics

Who: Dr Niall Madden (he/him)

How to greet: Niall ("Knee-al" `#StartsWithAName`)

From: 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.

This module involves a mix of lecture "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 before the Wednesday class.

Class times

	Mon	Tue	Wed	Thu	Fri
9 – 10				10	
10 – 11					6
11 – 12	8				✓
12 – 1	8		5	3	6
1 – 2			7		
2 – 3	6			-	
3 – 4	7	4	5		
4 – 5			✓		

We need to find some other times for labs.

Thanks to everyone who sent your time-table. Really sorry, but could you also complete this form at QR code above (also https://tallycal.com/p/4313600) ASAP, preferably by midday tomorrow (or now!). When doing do, indicate all times when you do not have a clash with a class – not just when it is inconvenient.

Overview of CS319

We'll use Canvas for

- Posting announcements (1 per week, usually);
- Posting grades
- Assignment uploads
- *Links* to slides and scripts from class.
  All materials will actually be hosted at
  `https://www.niallmadden.ie/2425-CS319`

Code (and there will be lots of it) can also be made available on a `git` repository at:
`https://github.com/niallmadden/2324-CS319` If interested, ask me!

Overview of CS319 — Text books

The notes for CS319 are largely self-contained. But some books will be VERY helpful. The reading list is at https://nuigalway.rl.talis.com/modules/cs319.html but will be updated. Key books include

- Scientific Computing with Case Studies (Diane O'Leary) https://epubs-siam-org.nuigalway.idm.oclc.org/doi/book/10.1137/9780898717723
- Practical C++ programming (Steve Oualline). https://search.library.nuigalway.ie/permalink/f/1pmb9lf/353GAL_ALMA_DS51566631000003626
- Think Python (Allen B. Downey) https://greenteapress.com/wp/think-python-3rd-edition

  3e.
- More will be added ...

The final grade for CS319 will be based on

- **Four programming assignments** (40%)
- a mid-semester open-book test (20%) (Week 6: TBC).
- a project and presentation (40%)

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

CS319: what and why

In CS319 we are primarily concerned with *three* 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 potential and limitations of the algorithm.
2. Implementing that algorithm: How to write the code! Not just that, we'll also learn how to code in C++. (Why C++? More on that later...)
3. Testing/verifying/validating the implementation (for correctness and efficiency).

CS319: what and why

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

It is NOT a “first course on programming”. You are expected to be proficient in at least one language. For most of you, that language is Python. For some it is Java. (Any others?)

The primary learning outcomes are that, by the end of the semester, you can honestly list

- “Skilled in scientific computing”
- “Can programme in C++”

on your CV.

There are various good candidates for a language with which to do Scientific Computing, including

- Python
- MATLAB/Octave
- C/C++
- Julia
- others?

Python: advantages

(i) Lots of great libraries, specially `NumPy`, `scipy`, and `matplotlib`;

(ii) Free!

(iii) Good IDEs/ notebooks.

(iv) Many people already have expertise.

(v) ...

Python: disadvantages

(i) Can be very slow.

(ii) Some of you are already quite expert, some less so.

(iii) ... not new (to you) ...

MATLAB/Octave: advantages

(i) Specifically designed for SciComp

(ii) Fast!

(iii) Lots of tools/libraries included; good at visualisation.

(iv) Excellent IDEs and notebook environment

(v) Everyone starts from the same place.

(vi) ...

MATLAB/Octave: disadvantages

(i) MATLAB is expensive

(ii) Skills not so transferable.

(iii) ...

C++: advantages

(i) Fast!
(ii) Valuable for your CV
(iii) Transferable skills (e.g, Arduino)
(iv) Everyone starts from the same place (right?).
(v) Free
(vi) ...

C++: disadvantages

(i) Very few libraries; no standard IDE
(ii) Steep learning curve
(iii) More time needed for studying the language
(iv) ...

So we'll make use of both C++ and Python:

- We'll learn the basics of coding in C++
- Our code will run very efficiently
- We'll import results into Python/Jupyter for further analysis.

You can use any Jupyter solution you like. To use the School one, watch out for an email from maths-sto@universityofgalway.ie then use the data provided to access https://cloudjupyter.universityofgalway.ie/

Or use Colab, Binder, etc.

Scientific Computing

Dianne O'Leary describes a **computational scientist** as someone whose focus is the intelligent development and use of 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.

Scientific Computing

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

For us, the major topics of CS319 will be

- **Computer representation of numbers**, as well as more complicated objects, such as vectors and matrices.
- Defining functions (of various types);
- Differentiation and integration of (mathematical) functions.
- **Root-finding and optimisation**
- Efficiency and complexity of algorithms (from an experimental/applied view).
- Solving linear systems by direct and iterative methods

A first example

In the first few weeks of the module, we'll emphasise C++ more that Scientific Computing.

First, though, we are going to study, without too much explanation, how to implement a simple algorithm in each of the three languages mentioned earlier: Python, MATLAB, and C++. We'll also estimate their (in)efficiency.

The problem is to write some code that will sum all the elements in a list, and report how long it took.

In each case, we'll take the simplest possible approach, and ignore that each of these languages has (somewhat built-in) functions to do this.

A first example

TimeAlg1.py

```
# Sum the elements of a list in Python
import time

N=10**8      # N=10^n
A = [1]*N
start = time.time()
s1=0;
for i in range(len(A)):
    s1+=A[i]
t1 = time.time() - start
print(f"N={N:6.0e}, error={s1-N}, time(s)={t1:6.2f}")
```

→ A is a list of 1's of length N.

A first example

TimeAlg1.m

```
% Sum the elements of a link in MATLAB/Octave
N = 10^8;    % N=10^n
A = ones(1,N);
start=tic;
s1 = 0;
for i=1:length(A)
   s1=s1+A(i);
end
t1=toc(start);
fprintf('N=%8.2e, error=%d, time(s)=%8.4f\n',...
        N, s1-N, t1)
```

A first example

C++

TimeAlg1.cpp

Finished here Wednesday

```cpp
#include <iostream>
#include <time.h>
#include <math.h>
int main() {
  int N=pow(10,8); // N=10^n
  double *A = new double [N];
  for (int i=0; i<N; i++)   A[i]=1.0;
  clock_t start=clock();
  double s1=0;
  for (int i=0; i<N; i++)   s1+=A[i];
  double num_clocks = (double)(clock()-start);
  double t1 = num_clocks/CLOCKS_PER_SEC;
  std::cout << "N=10^" << log10(N)
            << ", error=" << s1-N
            << ", time(s)=" << t1 << std::endl;
  return(0);
}
```