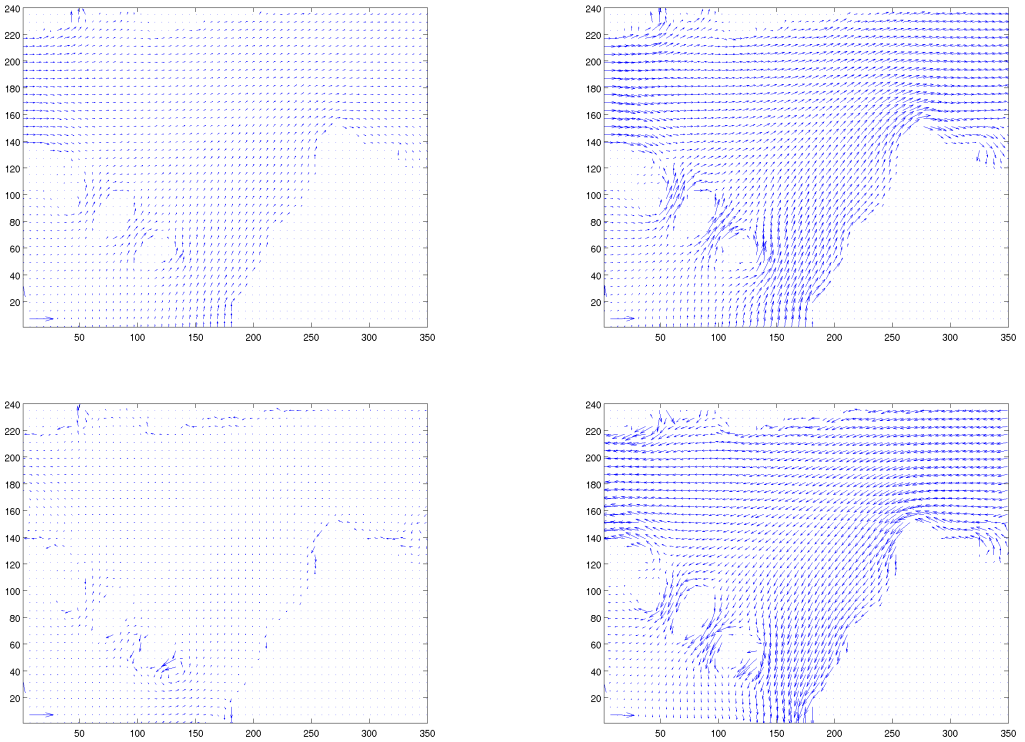
CS319: Scientific Computing

Introduction to CS319

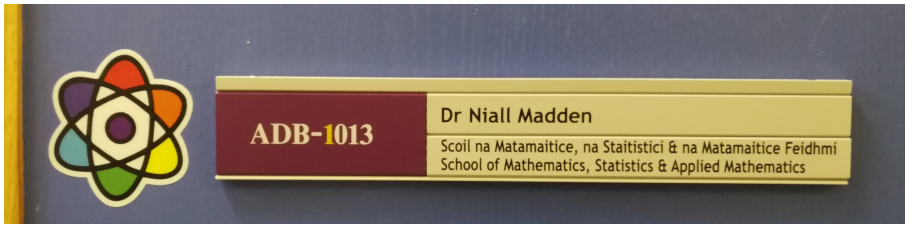
Dr Niall Madden

Week 1: 10 January, 2024



Outline

Lecturer details:



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); 4th Data Science, MSc Mathematics; Visiting Students (??).

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 start of the 9am class.

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/2324-CS319`

Code (and there will be lots of it) will be available on a `git` repository at: `https://github.com/niallmadden/2324-CS319`
Later I'll send you an invitation to it. If you didn't get it, send me an email.

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
- Learning MATLAB (Toby Driscoll). https://epubs-siam-org.nuigalway.idm.oclc.org/doi/book/10.1137/1.9780898717662
- Think Python (Allen B. Downey) https://greenteapress.com/wp/think-python-2e/
- 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 7).
- a project and presentation (40%)

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

	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~~ 3 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! (But in what language??)
3. Testing/verifying/validating the implementation.

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 outcome is that, by the end of the semester, you can honestly list "Skilled in scientific computing" on your CV.

We'll discuss 3 candidates for a language with which to do Scientific Computing:

- C/C++
- MATLAB/Octave
- Python

1. C++

Advantages: (1) Everyone starts at the same place!
(2) It is really useful! (3) Very fast.
Disadvantage: (1) Few libraries. (2) No standard IDE.
(3) Steep learning curve;
(4) Spend most of the time learning
C++, rather than SCcentific Computing.

We'll discuss 3 candidates for a language with which to do Scientific Computing:

- C/C++
- MATLAB/Octave
- Python

Finish here

10am.

MATLAB/Octave:
Advantages: (1) Everyone starts from the same place.
(2) Very fast. (3) Designed for Scientific Computing
(4) Lots of libraries; good at visualisation.

Disadvantages: (1) Expensive! (2) Obscure.

Python: Advantages - (1) Lots of great libraries: for us, NumPy and matplotlib; (2) Free, and with good IDEs/ notebooks.
Disadvantages: (1) Can be very slow. (2) Very different backgrounds...

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.

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.
- Visualisation of functions and data
- **Root-finding and optimisation**
- Data fitting, and least squares;
- 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

Python, Octave, and C++

In the first few weeks of the module, we'll use C++.
But first we are going to study, without too much explanation, how to implement a simple algorithm in each of these three languages, and 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.

TimeAlg1.py

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

N=10**7       # 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}")
```

TimeAlg1.m

```
% Sum the elements of a link in MATLAB/Octave
N = 10^6;    % 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)
```

TimeAlg1.cpp

```
#include <iostream>
#include <time.h>
#include <math.h>
int main() {
  int N=pow(10,6); // 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);
}
```