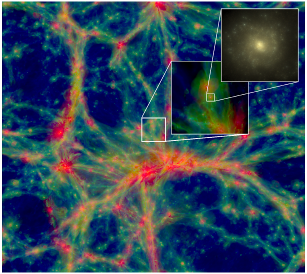
# Scientific Computing

Lecturer: Prof Tom Theuns
tom.theuns@durham.ac.uk
office: OCW 207

# About your lecturer:

I work at the Institute for Computational Cosmology



Computer simulation of the formation of galaxies. Background blue-green-red image shows intergalactic gas, coloured according to its temperature, inset zooms into a Milky Way-like galaxy. Shown volume is $3 \times 10^{24}$ m on a side. Figure from Schaye et al., '15. Want to know more, see the Eagle web site.

# Learning outcomes 'Scientific Computing'

- Learn how to use computer to solve complex (=realistic) problems
- Analyse physics to choose appropriate numerical algorithm
- Basic numerical methods and their implementation
  - differential equations
  - root finding: solutions for $f(x) = 0$
  - numerical integration
  - Monte Carlo methods & simulation
- Assessment via Python assignments in Jupyter notebooks,

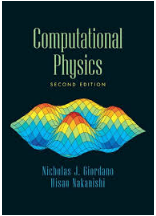
https://notebooks.dmaitre.phyip3.dur.ac.uk/miscada-sc/Teaching assistants are

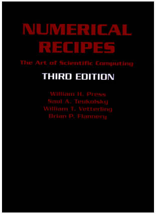
Dr Christian Arnold

Dr Emma Lofthouse

# Material:

- Course text book: Giordano & Nakanishi, "Computational physics" ($\sim$ 10 copies in library)
- Additional reading: Press et al., "Numerical Recipes"






- DUO online course notes/hand-outs: Core I: Statistics, Machine Learning, Scientific and High Performance Computing

## This course: Scientific computing

- Lecture on a given topic (e.g. radioactive decay), 2 topics per week
- Lab session on each topic, 2 lab sessions per week
- Course duration: 4 weeks
- Course work marked when Jupyter notebook session is submitted
- Feedback in following lecture
- Other topics in this module: statistics, machine learning, HPC

# Scientific computing:

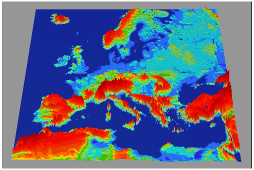
- Formulate a mathematical model for given problem
- Usually, analytical solution only possible once simplified
- Scientific computing to go beyond simplifications
    1. Analyse problem ('mathematical modelling')
    2. Choose and implement algorithm ('coding')
    3. Code verification ('debugging')
    4. Model validation/refinement ('experiment')
    5. Speed ('profile')
    6. Code documenting / upgrading (e.g. version control)

(This course)

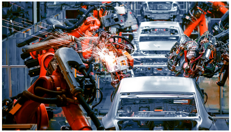
# Example: pendulum

- Mathematical model: $\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\frac{g}{l} \sin\theta$ . $l$: pendulum's length, $g$: gravitational acceleration, $\theta$ angle from vertical, $t$: time
- Analytical model (small angle approximation, $\sin(\theta) \approx \theta$) $\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\frac{g}{l}\theta$ .
  Solution: simple harmonic motion
- Numerical model: solve for $\theta$ not small, include air drag on pendulum bob, etc. No known analytical solution
- Code verification: test small-angle case
- Model verification: compare to real pendulum

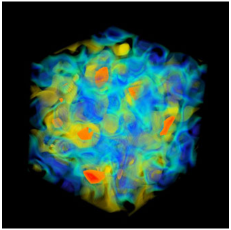
# Real-world examples



Weather forecasting

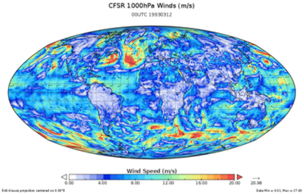


Improving efficiency of production chains



Fundamental physics

European HPC projects




Climate modelling

# Some vocabulary

- Solutions of mathematical models are functions

  e.g. for pendulum, the function $\theta(t)$

- Functions are maps:

$$\begin{array}{rcl} \text{set of input values} & \longrightarrow & \text{set of output values} \\ \text{domain} & \longrightarrow & \text{range}. \end{array}$$

  for pendulum: $t$ is input, $\theta$ is output

- An algorithm is a recipe for how to compute outputs given inputs
- An implementation codes the algorithm
- Computations evaluate functions
- In the digital world (computers), the domain and range are discrete, and the implementation terminates after a finite number of steps

## Types of errors

- Truncation error:
  Many functions are computed as a series, e.g., $\sin x = x - \frac{x^3}{3!} + \ldots$ or similar. Evaluation is limited to a finite number of terms.
- Finite precision error:
  Computer uses a finite number of bits to represent numbers. This leads to round-off errors and breaks commutativity of mathematical operations

  Example: $10^{30} + 1 - 10^{30} = 0$, and $a + b + c \neq a + c + b$
- Discretisation error:
  Computer approximates a smooth function with discrete steps. Accuracy improves with decreasing step-size.

Errors can accumulate, leading to instabilities. Poor implementation of algorithm yields incorrect answer.

# Course contents

in (): lecture/ws room, teaching assistant (TA), problem assistant

- 1-6: formatively assessed submit final notebook 6 days after lecture Sunday noon for Monday assignment, Thursday noon for Friday assignment
- 7+8: summatively assessed: paper submission **deadline November 15$^{\mathrm{th}}$**

1. Radioactive decay (TLC025, TA: Lofthouse, setter: Arnold)
2. Ballistic motion (Lecture OCW017, WS: CM001-3, TA: Lofthouse, setter: Arnold)
3. Harmonic motion (TLC025, TA: Lofthouse, setter, Arnold)
4. Chaos (Lecture OCW017, WS: CM001-3, TA: Lofthouse, setter: Lofthouse)
5. Root finding and integration (TLC025, TA: Arnold, setter: Lofthouse)
6. Random walks (Lecture OCW017, WS: CM001-3, TA: Arnold, setter: Lofthouse)
7. Cluster growth and percolation (TLC025, TA: Arnold, setter: Lofthouse)
8. Ising model and phase transitions (Lecture OCW017, WS: CM001-3, TA: Arnold, setter: Arnold)