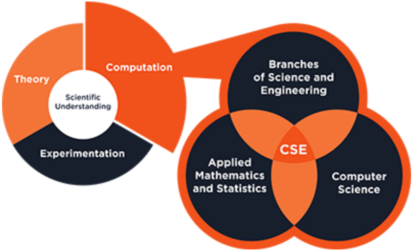
Intro to Computational Science and Engineering (CSE)

COMP1730/COMP6730 - Programming for scientists - Special Topic

Dr. Alberto F. Martín - alberto.f.martin@anu.edu.au
School of Computing, Australian National University, Canberra, 23th Oct 2023

Lecture outline - Introduction to CSE

- What is Computational Science and Engineering (CSE)?
- Main ingredients
- Application areas of CSE
- Two research grand challenges in CSE currently out-of-reach



The scientific method nowadays (three complementary pillars)

Theory (e.g. relativity, quantum mechanics, etc.)

- Mathematical models, theories, etc. (Domain expertise/knowledge lies here)
- Mathematical models do **NOT** have analytical/closed solutions in general (e.g. PDEs)

Experiments

- Grounded on observations of reality (e.g., weather balloons in weather forecasting)
- Too expensive (e.g., wind tunnel for full scale aeroplanes) or simply impossible (e.g., fusion energy, Mars mantle convection) in a vast array of cases

Computational Science and Engineering (CSE) - This lecture

- **Integrates applied mathematics, computer science, and branches of science/engineering in a single discipline**, e.g., computational biology, computational chemistry, computational fluid dynamics, computational geophysics, etc.
- Leverages **computational models** (e.g., discrete approximations resulting from advanced numerical methods), **algorithms**, **data**, **software** and **HPC** to tackle grand-challenges in science and engineering

Synergies:

Theory ↔ Experiments. Theory can predict reality/Experiments can validate theory
Theory ↔ CSE. Math models grounded on theory/Theory can validate computational models
Experiments ↔ CSE. Experiments can validate computational models/Computational models **can predict reality in complex scenarios!**

A taxonomy of mathematical models - Discrete vs Continuous

- Broadly speaking, there are two main types of mathematical models:
  - Discrete models
  - Continuous models
- **Discrete models**
  - In terms of a finite number of discrete entities and interactions among them
  - For example: atoms, molecules, etc.
  - Well-suited for computers (just FLOPs; computers can deal with this)
  - Examples: molecular dynamics, chemical reactions
- **Continuous models** (e.g., Partial Differential Equations - PDEs)
  - Encode the laws of nature using 1st physics principles (e.g., motion Newton's laws)
  - Involve continuous functions on an infinite set (e.g., the real line)
  - Expressed in terms of **integrals** and **derivatives** of functions
  - Computers don't know anything about functions, derivatives, or integrals!
  - We humans have to transform continuous models into discrete ones (i.e., FLOPs)

Example of continuous models: Partial Differential Equations (PDEs)

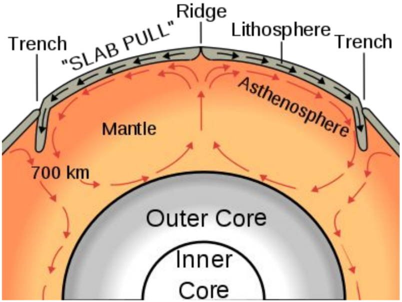
Earth's Mantle convection (*experiments not possible*)

Can be modelled as a PDE system: find **fluid velocity** $\boldsymbol{u}(\mathbf{x}, t)$, **pressure** $p(\mathbf{x}, t)$, and **temperature** $T(\mathbf{x}, t)$ s.t.:

$$-\nabla \cdot (\eta(\boldsymbol{u}, T)(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)) + \nabla p = \mathrm{Ra} T \boldsymbol{e_r}$$

$$\nabla \cdot \boldsymbol{u} = 0$$

$$\partial_t T + \boldsymbol{u} \cdot \nabla T - \nabla^2 T = \gamma$$



Click here for video animations of mantle convection simulations

(Source: ASPECT geodynamics scientific software)

Numerical methods: from continuous to discrete models

- We use numerical methods to transform continuous problems into (**computer-solvable!**) discrete problems
- Example: modelling heat conduction in a 1D metal bar

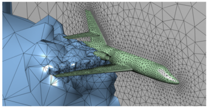
Laplace differential equation (continuous model)	**Linear system (discrete model)**
Find temperature $u(x)$ such that: $-\partial_x(\kappa(x)[\partial_x u(x)]) = f(x) \text{ within bar}$ (boundary conditions omitted for simplicity)	Find solution vector $U \in \mathbb{R}^n$ such that: $AU = F$ with $A \in \mathbb{R}^{n\times n}$ (matrix) and $F \in \mathbb{R}^n$ (vector)

- This comes at a price: **numerical errors and biases** (!!!)
- The expectation is that the more resolution in the discrete model, the higher the computational demands and the lower the error
- Mathematicians (**numerical analysts**) can prove bounds for these errors thus certifying the robustness and accuracy of the discrete models

Example: FEM simulation pipeline steps (common approach)

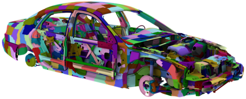
1. Unstructured mesh generation

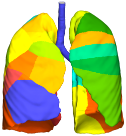
Delaunay triangulations mainstream



2. Mesh partition

Graph-based algorithms mainstream

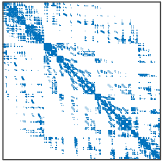




3. Discrete system assembly

Involves numerical integration on elements
Embarrassingly (trivially) parallel process

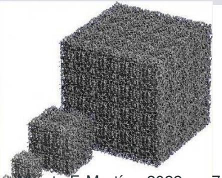
$$AU = F$$



4. Discrete system solvers

Significance of **algorithmically scalable** solvers (FLOPs/mem demands linearly bounded with resolution)

Multilevel methods mainstream for discrete PDEs (Multigrid, Multilevel Domain Decomposition)



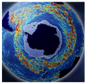
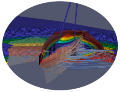
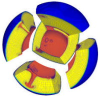
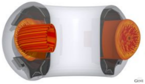
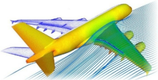
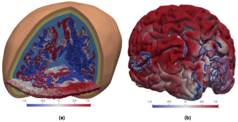
- Objective: improve the state-of-the-art in **computational models**, **algorithms**, and **software** to push the boundaries of what is currently achievable in CSE
- **Strong potential**: simulate out of reach problems, more precise predictive CSE, improved scientific knowledge, revolutionize decision-making across science, technology and society

Main research areas

- Mathematical modelling
- Numerical methods (**discretization**, **solvers**)
- Data assimilation (e.g., **machine learning**)
- **HPC** (parallel software/hardware innovations)

Application areas (examples)

- Geophysics
- Nuclear fusion
- Aeronautics
- Personalized medicine (brain/heart)
- Nanoscience, Smart manufacturing, (large) Etc.

Synergy among HPC and CSE is crucial

- We already find ourselves in the **Exascale** era ($\mathcal{O}(10^{18})$ FLOPs/s peak)
- **Frontier**: 1st Exascale supercomputer (Oak Ridge US National Labs) ($\sim$10M cores, 1.1EFLOPs/s, ranked #1 Jun, 2023 Top500 list)

- Performance boost mostly based on adding hardware parallelism (e.g., higher #cores/CPU) and heterogeneous hardware (CPUs, GPUs, ...)
- To exploit such vast concurrency is a **formidable task** for CSE (breakthroughs in scalable algorithms and software innovations)

High quality scientific software is crucial for CSE

- Development of high quality, generally applicable, and publicly available high performance scientific software is key for CSE as a discipline
- Vast array of high quality open source CSE software available in the public domain, e.g.: TRILINOS, PETSc, FENICs, Firedrake, OpenFOAM, deal.II, (and a **large** etc.)
- From a research point of view, scientific software is a **key component** (increases impact, scientific reproducibility, builds a community around your research, etc.)
- I am one of the leaders of the Gridap.jl scientific software ecosystem of Julia packages. You can learn more about these efforts in my webpage and references therein

An out-of-reach example problem: full scale simulation of turbulent flows

- Turbulent flows (literally all around us) are complex phenomena that possess a set of features that render their full scale simulation out-of-reach computationally even with state-of-the-art algorithms and the most powerful Exascale supercomputers
- They are generally **3D**, **multi-scale** (in time and space), **mixing**, **unsteady**, and **highly-nonlinear** physical phenomena
- Typically modelled as a Continuous by the Navier-Stokes equations: Find **fluid velocity** $\boldsymbol{u}(\mathbf{x}, t)$, and **pressure** $p(\mathbf{x}, t)$ s.t.

$$\begin{aligned} \partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} &= -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u} \quad && \text{in } \Omega \times (0, T] \\ \nabla \cdot \boldsymbol{u} &= 0 && \text{in } \Omega \times (0, T] \end{aligned}$$

- Established methods in CSE include (ordered by decreasing accuracy, decreasing computational demands): DNS, LES, RANS
- Example: M. Hosseini, R. Vinuesa, et. al., *Turbulent flow around a wing profile, a direct numerical simulation*. V0078, APS Gallery of Fluid Motion, 2015. Available at YouTube here

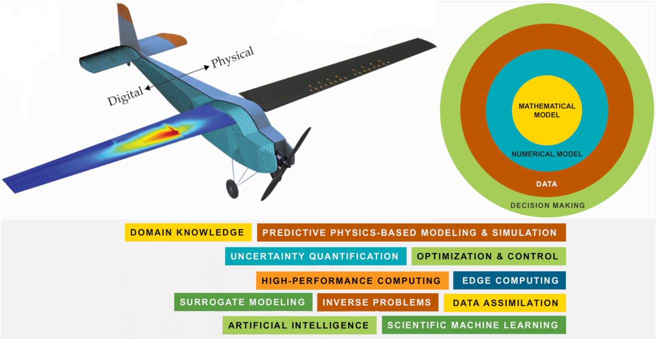
Full scale simulation of turbulent flows and deep learning

- Last years have seen a tremendous surge in research on deep learning techniques to enhance the fidelity of turbulent flow simulations and/or reduce their computational demands (e.g., via reduce-order modelling)
- See the following survey paper on advances on this field:

  R. Vinuesa, S. L. Brunton. *Enhancing computational fluid dynamics with machine learning.* Nature Computational Science, 2, pp 358–366, 2022. Available here

Another out-of-reach problem: Digital Twins

- Term first coined by NASA in the 60s (as part of Apollo mission)
- A **Digital Twin** is an evolving virtual representation of an object, system or organ that spans its lifecycle, is updated from real-time data, and uses simulation, ML, and reasoning to aid in decision-making



Source: SIAM Supercomputing Spotlights Talk by Prof. Karen Wilcox (UT Austin)

"How HPC is Personalizing the Future of Complex Systems". Available at YouTube here

CSE-related courses and workshops at ANU

Some CSE-related courses organized by the School of Computing (non-exhaustive list):

- COMP2710 - Numerical Computing with Julia (S2/2023)
- COMP3320 - High Performance Scientific Computation (S2/2023)
- COMP4300 - Parallel Systems (S1/2024)

For mathematically oriented students:

- MATH3512 - Matrix Computations
- MATH3511 - Scientific Computing
- MATH3514 - Numerical Optimisation
- MATH3349 - Numerical methods for time-dependent PDEs

I am organizing a hands-on workshop at ANU (late Nov, 2023) on finite element methods for PDEs using the Gridap.jl Julia ecosystem of packages