Computational Astrophysics

Violeta González Pérez

2022

Introduction to Computational Astrophysics

- Historical overview of Computational Astrophysics
- Supercomputers and Computational Astrophysics today
- Introduction to Numerical Modelling

Introduction to Computational Astrophysics

- Historical overview of Computational Astrophysics
- Supercomputers and Computational Astrophysics today
- Introduction to Numerical Modelling

A bit of the history of computation

Abacus $\rightarrow$ Pascal calculator to add numbers in the XVII century $\rightarrow$
Babbage designs and partial machines $+$ Ada Lovelace precursor for coding (1830s) $\rightarrow$
Human computers: Astronomical Distances (Henrietta Leavitt), Manhatan Project, Decoding (Enigma at Bletchley Park)
WWII: strong development of hardware (in secret), Mark I, Z1, etc., and concepts like the Turning test
At the end of WWII: machines where faster than humans, but it could take days to program a calculation of seconds (Hardware $\sim$ Software).
ENIAC one of the first commercial machine, programmed by ENIAC 6: Kay McNulty, Betty Jennings, Betty Snyder, Marlyn Meltzer, Fran Bilas, and Ruth Lichterman.
Grace Hopper, programmer for MarkI and II: created the concept of subroutines, documentation, the first compiler (1951) and one of the first high-level programming languages (COBOL).

The first cosmological simulation?

Erik Holmberg

THE ASTROPHYSICAL JOURNAL

AN INTERNATIONAL REVIEW OF SPECTROSCOPY AND ASTRONOMICAL PHYSICS

VOLUME 94 NOVEMBER 1941 NUMBER 3

ON THE CLUSTERING TENDENCIES AMONG THE NEBULAE

II. A STUDY OF ENCOUNTERS BETWEEN LABORATORY MODELS OF STELLAR SYSTEMS BY A NEW INTEGRATION PROCEDURE

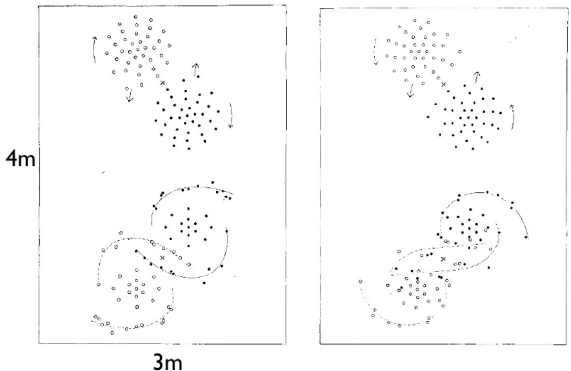
ERIK HOLMBERG

ABSTRACT

The first cosmological simulation: using light

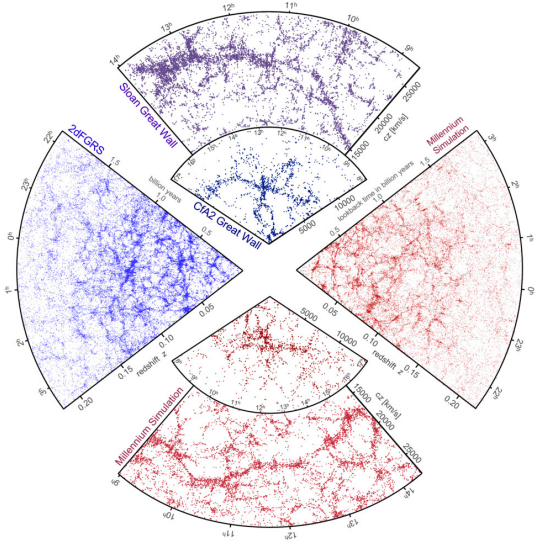
N = 2x 37

- replacing gravity by light (same $1/r^2$ law)
- formation of tidal features



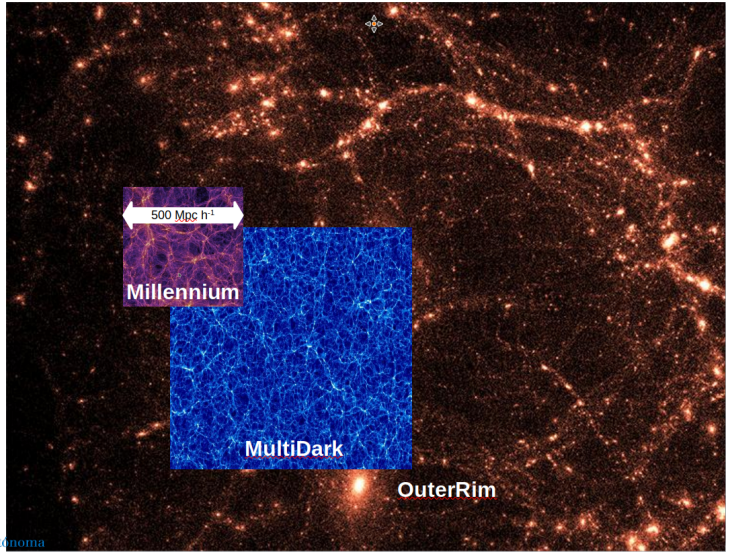
It took 20 years to do somehting similar with a computer.

Historical computational Astrophysics



Springel+2006

Towards bigger and more detailed simulations



Introduction to Computational Astrophysics

- Historical overview of Computational Astrophysics
- Supercomputers and Computational Astrophysics today
- Introduction to Numerical Modelling

Supercomputer speeds: FLOPS

The speed of computers is measured in terms of the number of floating-point operations per second they can perform.

Floating-point operations per second = FLOP, flops or flop/s

How fast is your machine?

1. Create an account in GitHub.
2. Go to `https://github.com/AMDmi3/flops` and **Fork** the repository.
3. Install git in your computer.
4. In your computer:

```
$ git clone git@github.com:[gituser]/flops.git
$ cd flops
$ make
$ ./flops
```

Quick introduction to `git`

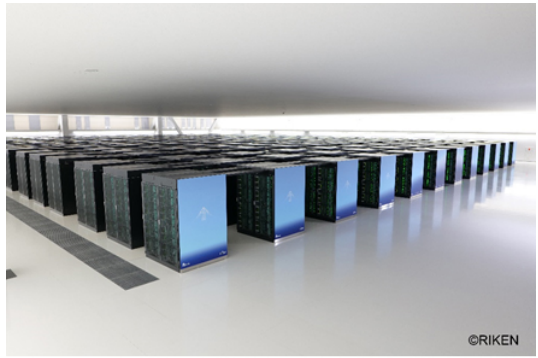
`git` is a version control tool that allow us to work on different copies of a code allowing for collaboration and controlled development. Here there are the basic commands you'll need day to day:

1. Add files to a repository:
   ```
   $ git add [file]
   ```
2. Save your changes locally:
   ```
   $ git commit -am "[Description of changes]"
   ```
3. Save your changes in GitHub:
   ```
   $ git push
   ```

The fastest supercomputer in the world

The current *fastest* supercomputer according to Top500 (11/2021):

- Fugaku, "Mount Fuji", in Japan:
  - 7,630,848 cores
  - Total memory: 4.85 PiB; Memory Bandwidth: 163 PB/s
  - Speed: 442 Pflops = 442 10^9Mflops = 442 10^{15}flops
  - Compilers: Fortran2008, Fortran2018, C11, C++14, C++17, GNU, OpenMP 4.5, OpenMP 5.0, Java
  - Script Language: Python + Numpy + Scipy, Ruby
  - Operative System: Red Hat Enterprise Linux 8



Fugaku is 3x faster than the 2nd one, Summit (USA).

The fastest supercomputer in Spain

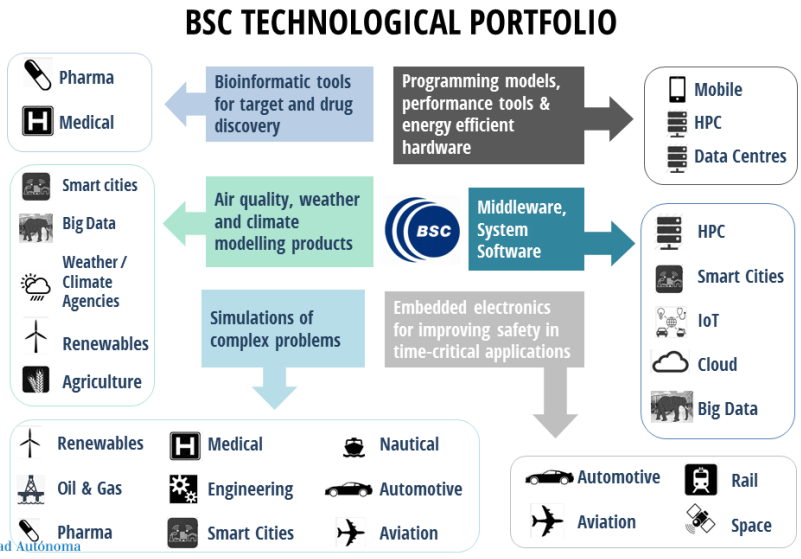
The current *fastest* supercomputer in Spain (# 73 Top500 11/2021):

- Marenostrum (at Torre Girona Chapel, Barcelona):
  - 153,216 cores
  - Memory: 41,664 GB
  - Speed: 7 Pflops
  - Operative System: Suse Linux

Question

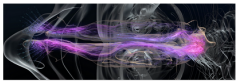
Do you think you interact with supercomputers in your daily life?

Technological applications of supercomputers



Scientific applications of supercomputers

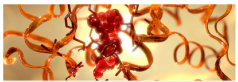

ATMOSPHERIC COMPOSITION

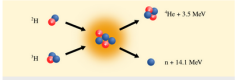

BIG DATA


EDUCATION

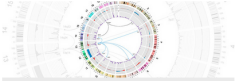

ENGINEERING SIMULATIONS

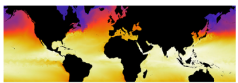

BIOINFORMATICS

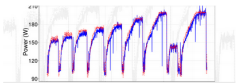

BIOMECHANICS



FUSION

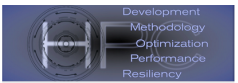

GENOMICS


CLIMATE PREDICTION



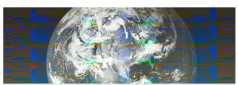
CLOUD COMPUTING

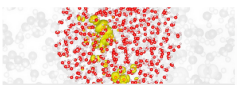

GEOPHYSICS

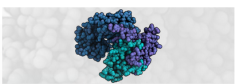


HPC SOFTWARE ENGINEERING


COGNITIVE COMPUTING


COMPUTATIONAL EARTH SCIENCE


MATERIAL SCIENCE

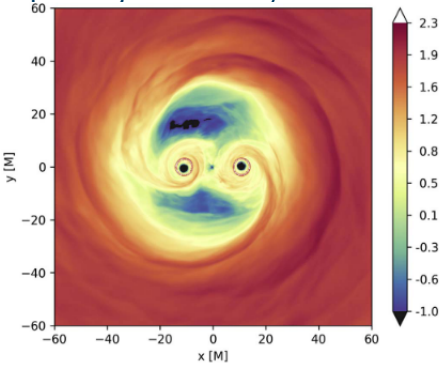

MOLECULAR MODELING

Some research areas at the Barcelona Supercomputer Center

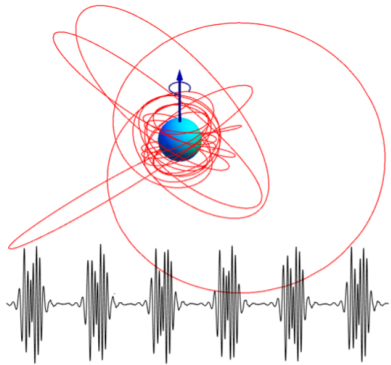
Computational Astrophysics and Cosmology today: within a galaxy

Stars and interstellar medium: astrosismology, exoplanet search, the origin of gamma burst and other very energetic and mysterius events.

Compact objects: Gravitational waves have open a window that will allow us understang compact objects and maybe test different cosmological models.



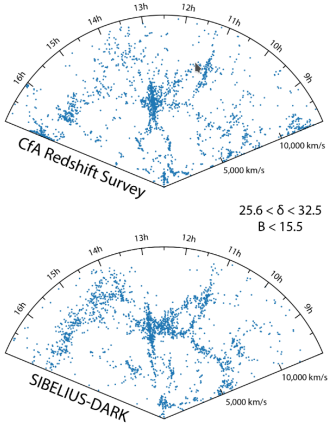
d'Ascoli+2018



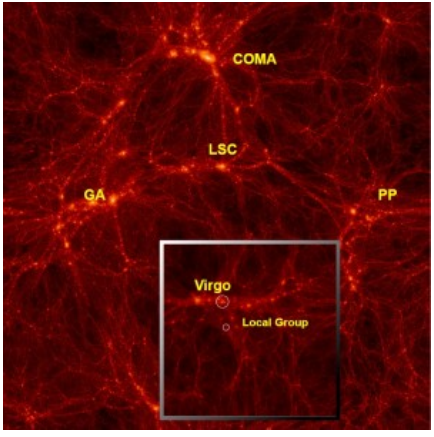
At UAM: LIGO colaboration

Computational Astrophysics and Cosmology today: Local group

Milky Way and the Local group: constraining Dark Matter and the history and future of our galaxy.



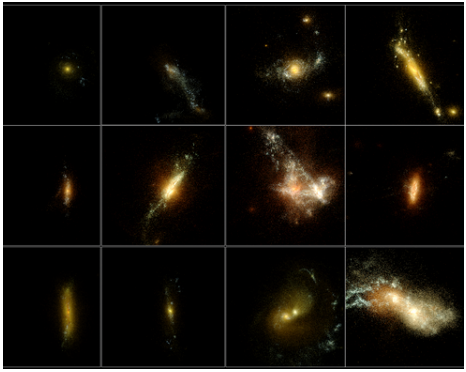
McAlpine+2022



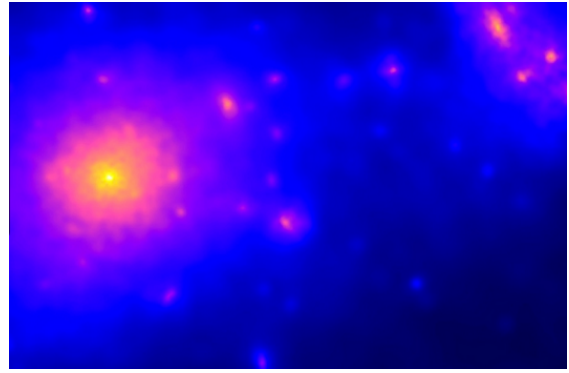
At UAM: the CLUES simulation

Computational Astrophysics and Cosmology today: Galaxies and gas

From individual galaxies to clusters of galaxies: how do galaxies form and evolve, constrains to cosmological models.



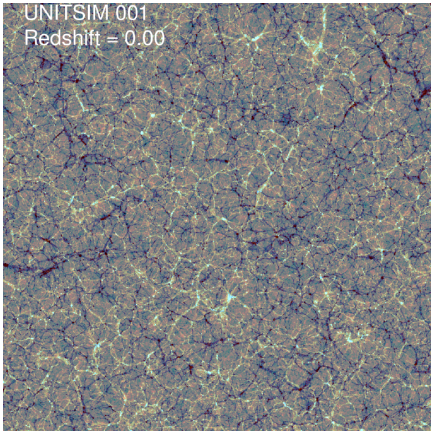
At UAM:Vela simulations



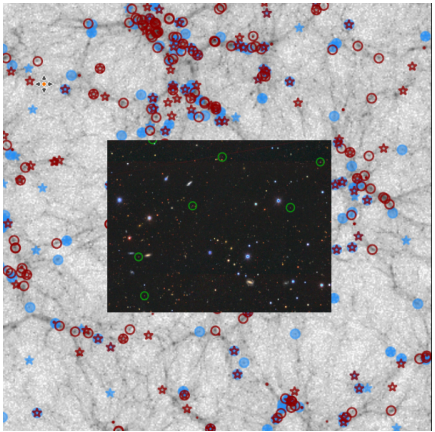
At UAM: The 300 project

Computational Astrophysics and Cosmology today: LSS

Large Scale Structure: galaxy evolution and cosmology.



At UAM: UNIT simulation



At UAM: SAMs

Introduction to Computational Astrophysics

- Historical overview of Computational Astrophysics
- Supercomputers and Computational Astrophysics today
- Introduction to Numerical Modelling

Numerical simulations

Nature's definition:
A numerical simulation is a calculation that is run on a computer following a program that implements a mathematical model for a physical system. Numerical simulations are required to study the behaviour of systems whose mathematical models are too complex to provide analytical solutions, as in most nonlinear systems.

Numerical simulations: an aid to understand physical phenomenon

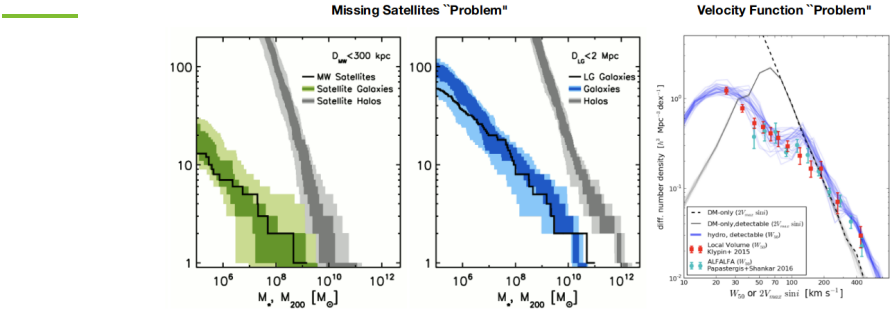
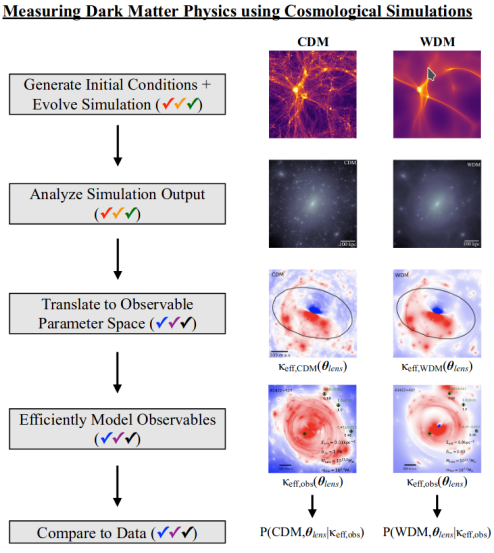


Figure 3: Examples where a proper comparison between simulations and observations has helped resolve purported "problems" in CDM. *Left and middle panels*: modeling galaxy formation physics on top of N-body simulations helps resolve the missing satellites problem. Galaxy formation lowers the predicted number of subhalos in CDM (shown in gray) to the number of predicted luminous dwarf galaxies (green/blue), which are in good agreement with observations of the MW and the Local Volume [150]. *Right*: in addition to galaxy formation physics, the translation from circular velocity function (solid gray and dashed lines) to HI-width function (blue lines) helps explain the shallow slope in the observed HI-width velocity function [37].

Banerjee+2022

Steps from a physical phenomenon to explanation



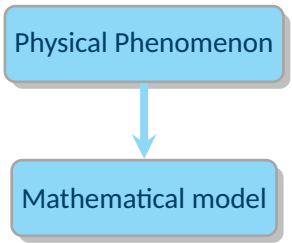
Banerjee+2022

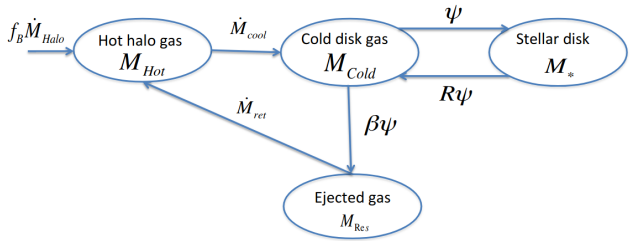
What is the physical process we aim to understand?

Physical Phenomenon

https://www.legacysurvey.org/

How can we describe analytically the phenomenon?

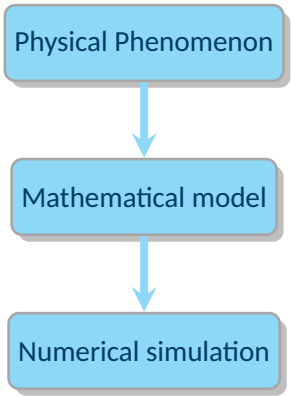




$$\begin{pmatrix} \dot{M}_{\rm DM} \\ \dot{M}_{\rm hot} \\ \dot{M}_{\rm cold} \\ \dot{M}_{\star} \\ \dot{M}_{\rm res} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\tau_{\rm ret}} \\ 0 & 0 & \frac{-1(1-R+\beta)}{\tau_\star} & 0 & 0 \\ 0 & 0 & \frac{(1-R)}{\tau_\star} & 0 & 0 \\ 0 & 0 & \frac{\beta}{\tau_\star} & 0 & \frac{-1}{\tau_{\rm ret}} \end{pmatrix} \begin{pmatrix} M_{\rm DM} \\ M_{\rm hot} \\ M_{\rm cold} \\ M_{\star} \\ M_{\rm res} \end{pmatrix} + \begin{pmatrix} (1-f_{\rm B})\dot{M}_{\rm H} \\ f_{\rm B}\dot{M}_{\rm H} - \dot{M}_{\rm infall} \\ \dot{M}_{\rm infall} \\ 0 \\ 0 \end{pmatrix}$$

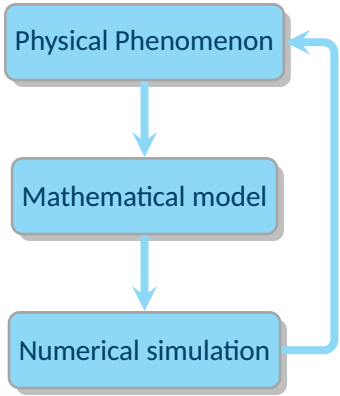
Mitchell+2017

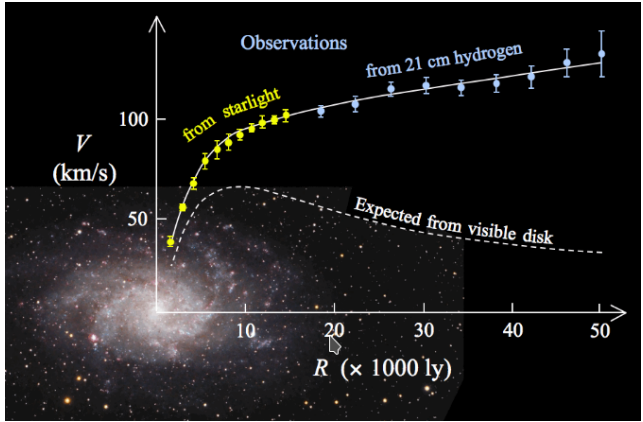
Numerical implementation of the analytical model



Millennium DM sim + Galform + Daniel Farrow's code

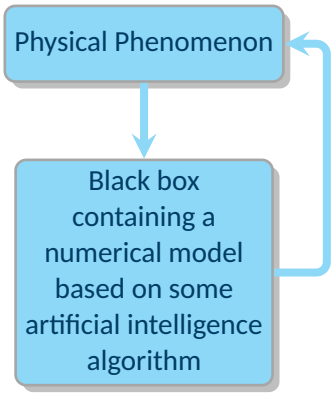
How do the results compare to the observable?

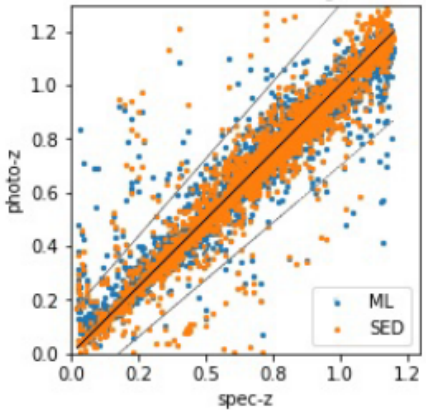




M33 rotation curve

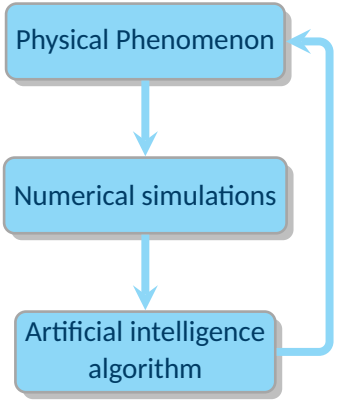
This loop can be different when using artificial intelligence

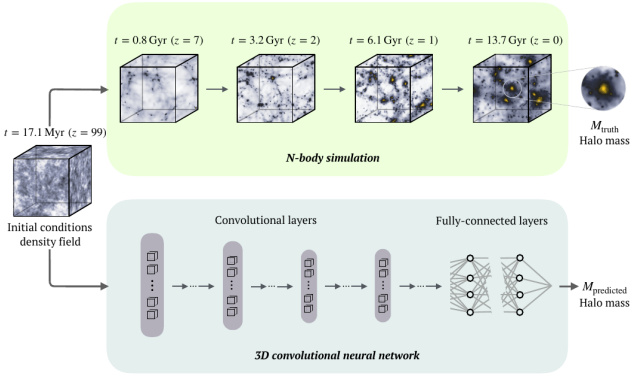




Brescia+2021

This loop can be different when using artificial intelligence





Lucie-Smith+2021

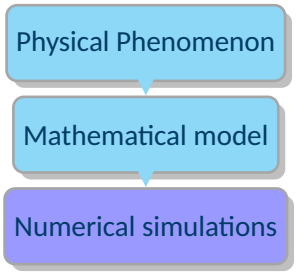
Numerical simulations: the heart of this course

The idea of the end course **project** is for you to work through this pipeline either by

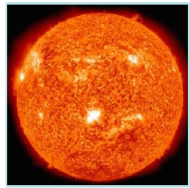
- Working with publicly available code on a physical phenomenon that interest you.
- Writing your own code to tackle a given problem.

In practice (from Astrophysical Recepies):

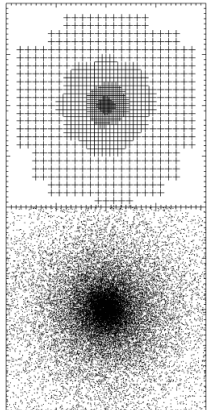
1. Start with a scientific question
2. Identify the minimal physics needed to address the question
3. Find an example script that resembles the target problem as closely as possible
4. Rewrite that script to make it solve our problem.

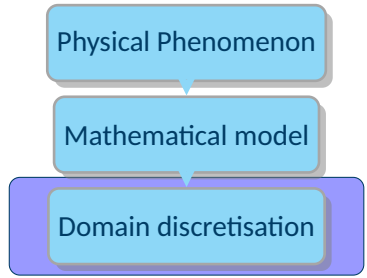


Numerical simulations: Domain discretisation



- grid introduction
- particle sampling
- ...





Question

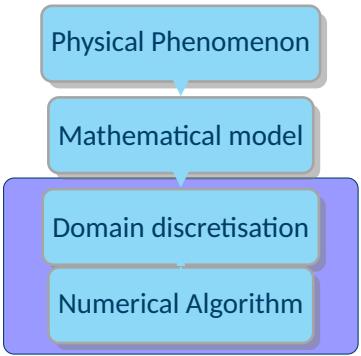
How do you plot $sin(x)$?

Numerical simulations: Numerical Algorithms

We need to find numerical algorithms that deal with our mathematical model, which is a set of analytical equations. For example,

$$I = \int_a^b f(x)\mathrm{d}x \Rightarrow \sum_{i=1}^{N-1} \frac{f(x_{i+1}) + f(x_i)}{2}(x_{i+1} - x_i)$$

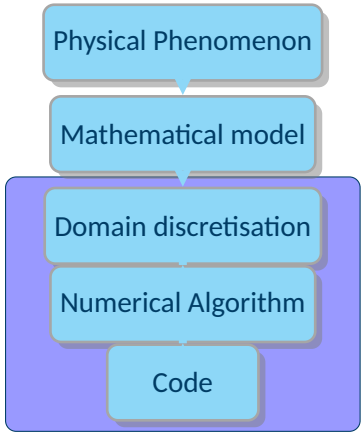
In most cases for Computational Astrophysics we will need to solve differential equations. Different discretizations will require different integration schemes.



Numerical simulations: coding

Code = human readible instructions for a machine.

1. **High-level programming languages:** They require compilation to become ”machine code”. The compilation step makes them faster.
   C, C++, Fortran, ...
2. **Script programming languages:** The scripts are interpreted while running the code, making them slower. They are easier to code and provide visualisations capabilities.
   Python, JavaScirpt, ...
3. **Hybrid programming languages:** Written like a script language, including interactive features, but being compiled in the background. Julia

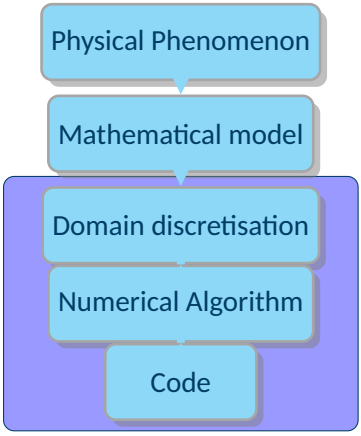


Numerical simulations: compile and run

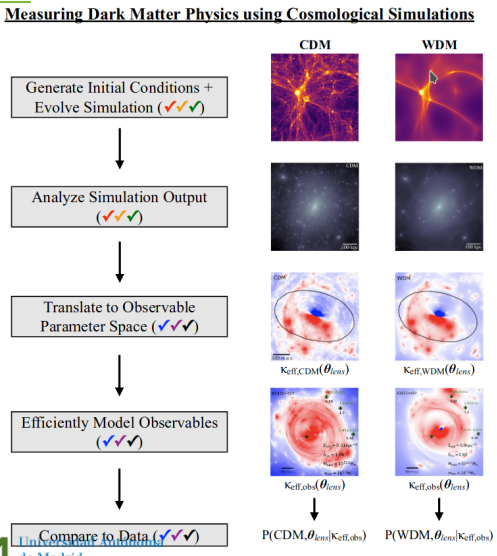
Find a supercomputer, (compile) and run.

Use example for C:

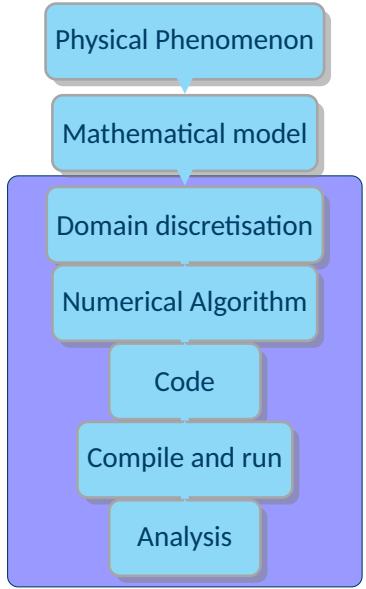
```
$ make
$ run
```



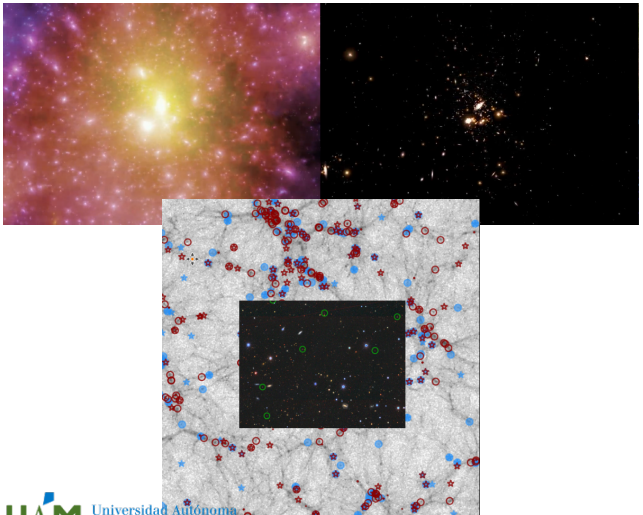
Numerical simulations: Analysis

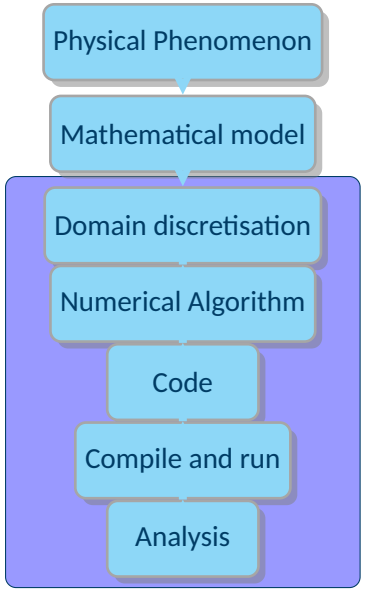


Banerjee+2022

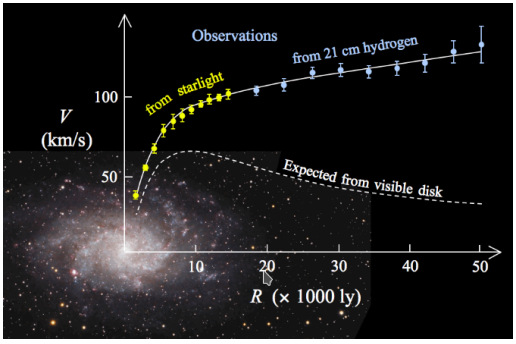


Numerical simulations: Analysis

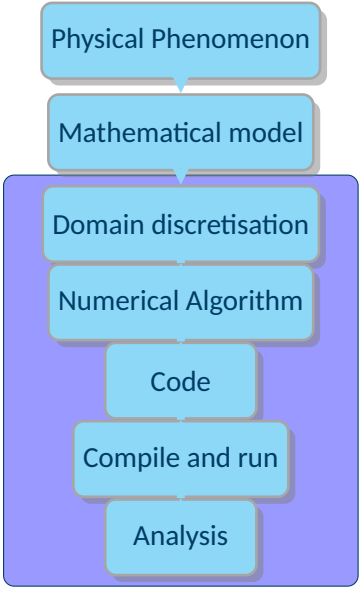




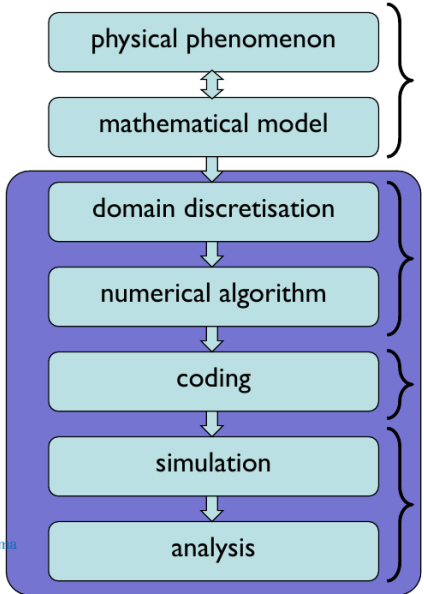
Numerical simulations: Before claiming new physics



However, when the simulations differ from the observations, we need to check all the different steps involved in the process.

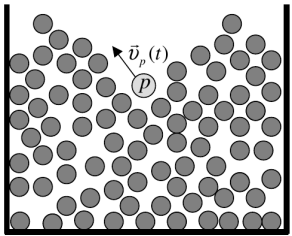


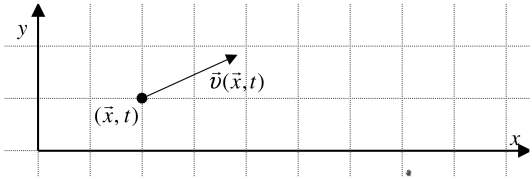
Numerical simulations: the heart of this course



- lecture "astrophysical processes"
- all remaining lectures
- practical lessons
- exercises & projects

Fundamentally different approaches for describing a fluid





- **Lagrange description:** describes the evolution of a fluid by following individual particles. We could do on equal mass particles. This method is computationally expensive.

- **Eulerian description:** follows the evolution of the properties of a region of the space. We could implement it in equal volume cells. It describes the field (volume element) instead of individual particles and thus, can be faster than the Lagrangian approach.

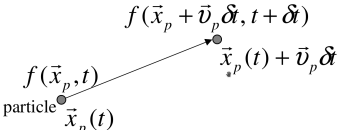
There are codes, like AREPO that combine these two approaches.

Moving derivative, $\mathrm{D}/\mathrm{D}t$

or hydrodynamical derivate, material derivative, total, Lagrange, Stokes, etc.

Let's consider an Eularian quantity $f(\vec{x}(t), t)$. The temporal change of a particle p:

$$\frac{\mathrm{D}f(\vec{x}(t), t)}{\mathrm{D}t} = \lim_{\delta t \to 0} \frac{f(\vec{x} + \vec{v_p}\delta t, t + \delta t) - f(\vec{x}, t)}{\delta t}$$



Taylor expansion around $(\vec{x_p}, t)$, with $\vec{v_p}\delta t = \delta\vec{x}$:

$$f(\vec{x_p}+\vec{v_p}\delta t, t+\delta t) = f(\vec{x_p}, t)+\delta t\frac{\partial f(\vec{x_p}, t)}{\partial t}+\delta\vec{x_p}\cdot\nabla f(\vec{x_p}, t)+\mathcal{O}(\delta^2)$$

$$\frac{\mathrm{D}f}{\mathrm{D}t} = \frac{\partial f}{\partial t} + \vec{v_p}\cdot\nabla f \quad \xrightarrow{\text{in general}} \quad \frac{\mathrm{D}}{\mathrm{D}t} \equiv \frac{\partial}{\partial t} + \vec{v_p}\cdot\nabla \quad \xrightarrow{\text{spherical sim.}} \quad \frac{\mathrm{D}}{\mathrm{D}t} \equiv \frac{\partial}{\partial t} + v\frac{\partial}{\partial r}$$

$\mathrm{D}/\mathrm{D}t$ is the temporal derivative of a property, following the particle ($\sim$ 'Lagrangian' concept), expressed in terms of Eulerian quantities. In this way, we can apply the laws of conservation on an Euler reference system.

Differential equations with two different domain discretisations

- domain discretisation

L. P. Euler
1707-1783

vs.

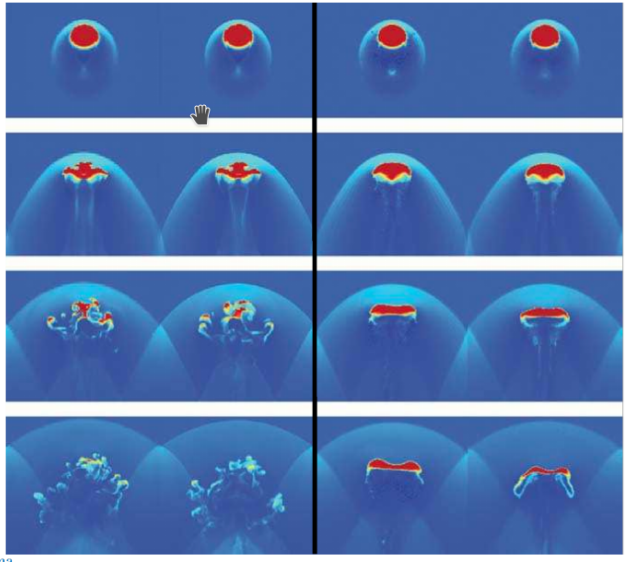
J.-L. Lagrange
1736-1813

$\partial/\partial t$ = rate of change at a fixed point in space

D/Dt = rate of change following a mass element

$\frac{\partial \rho}{\partial t} + v_i \frac{\partial \rho}{\partial x_i} = -\rho \frac{\partial v_i}{\partial x_i}$	mass conservation	$\frac{D\rho}{Dt} = -\rho \frac{\partial v_i}{\partial x_i}$
$\frac{\partial v_i}{\partial t} + v_i \frac{\partial v_j}{\partial v_i} = -\frac{1}{\rho} \frac{\partial p}{\partial x_i}$	momentum conservation	$\frac{Dv_i}{Dt} = -\frac{1}{\rho} \frac{\partial p}{\partial x_i}$
$\frac{\partial e}{\partial t} + v_i \frac{\partial e}{\partial x_i} = -\frac{p}{\rho} \frac{\partial v_i}{\partial x_i}$	energy conservation	$\frac{De}{Dt} = -\frac{p}{\rho} \frac{\partial v_i}{\partial x_i}$

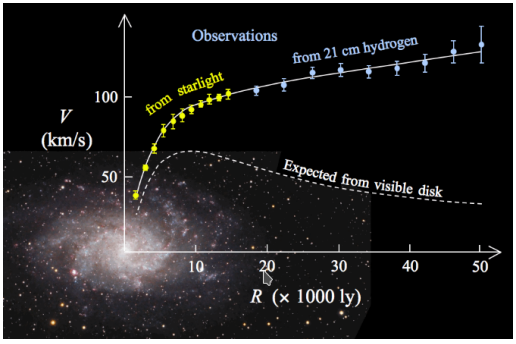
Differential equations with two different domain discretisations

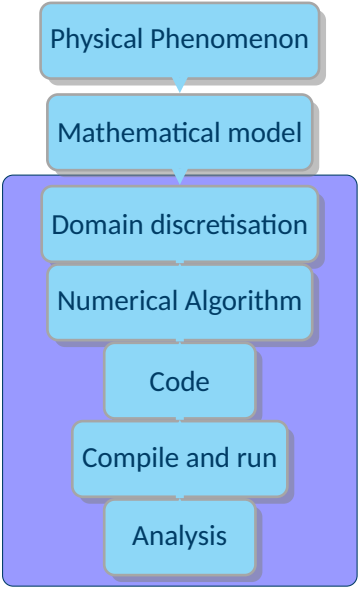


Agertz et al. (2007)

grid-based methods vs. particle-based methods

Numerical simulations: Before claiming new physics





However, when the simulations differ from the observations, we need to check all the different steps involved in the process.