Numerical methods for physics

Sebastien Charnoz and Andrea Ciardi

Computational physics

Computational physics aims at finding numerical solution to problems for which a *quantitative theory* already exists.

What does a computational physicist do:
→ develops/choses a quantitative theory (physical model)
→ develops/choses appropriate algorithms (numerical analysis)
→ implements the algorithms (computer code)
→ runs *numerical simulations* and *interprets* the results

Physical model

$$\frac{\partial \rho}{\partial t} + \nabla \cdot [\rho \mathbf{v}] = 0$$

$$\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot [\rho \mathbf{v}\mathbf{v} - \mathbf{B}\mathbf{B} + \mathrm{P}^*] = 0$$

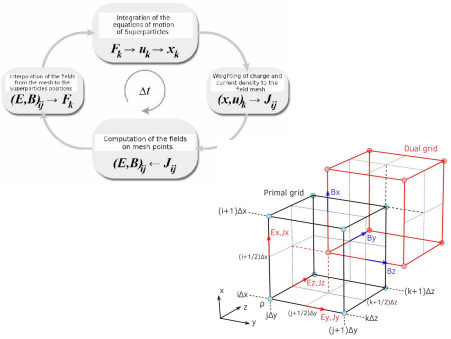
$$\frac{\partial E}{\partial t} + \nabla \cdot [(E + P^*)\mathbf{v} - \mathbf{B}\,(\mathbf{B} \cdot \mathbf{v})] = 0$$

$$\frac{\partial \mathbf{B}}{\partial t} - \nabla \times (\mathbf{v} \times \mathbf{B}) = 0$$

$$P^* = P + \frac{\mathbf{B} \cdot \mathbf{B}}{2}$$

$$E = P/(\gamma - 1) + \frac{\rho(\mathbf{v} \cdot \mathbf{v})}{2} + \frac{\mathbf{B} \cdot \mathbf{B}}{2}$$

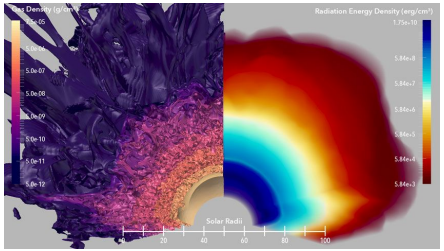
Algorithms and Numerical Methods



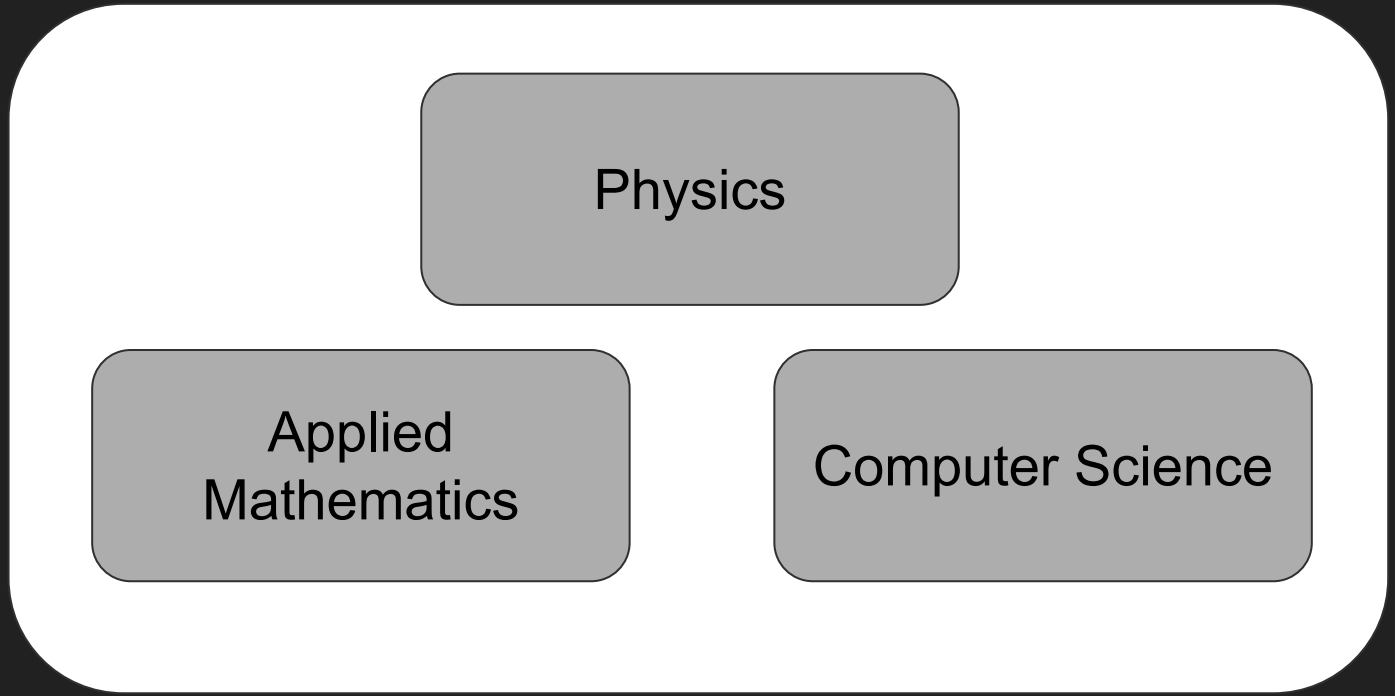
Computer code

```
!-----------------------main program
  use devObject
  implicit none
!-----------------------Executables
  call open_devObjects
  !
  call usersub
  !
  call close_devObjects
  end
!-----------------------end of main

subroutine usersub
   use devObject
   implicit none

   integer,parameter:: size=1024
   type(devVar) dv1,dv2,dv3,dv4

   real arld1(size),arld2(size),arld3(size), arldresult(size)

   call random_number(arld1)
   call random_number(arld2)
   call random_number(arld3)

   dv1=allocate_dv('real',size)
   dv2=allocate_dv('real',size)
   dv3=allocate_dv('real',size)

   call transfer_r4(arld1,dv1,.true.)
   call transfer_r4(arld2,dv2,.true.)
   call transfer_r4(arld3,dv3,.true.)

   !pointwise multiplication division addition subtraction test
   dv4=(3.141592*dv1*(dv2+dv1)*(dv1-(.5514*dv1))*dv2+(-.24))/dv1

   call transfer_r4(arldresult,dv4,.false.)

   call deallocate_dv(dv1)
   call deallocate_dv(dv2)
   call deallocate_dv(dv3)
   call deallocate_dv(dv4)
end subroutine usersub
```

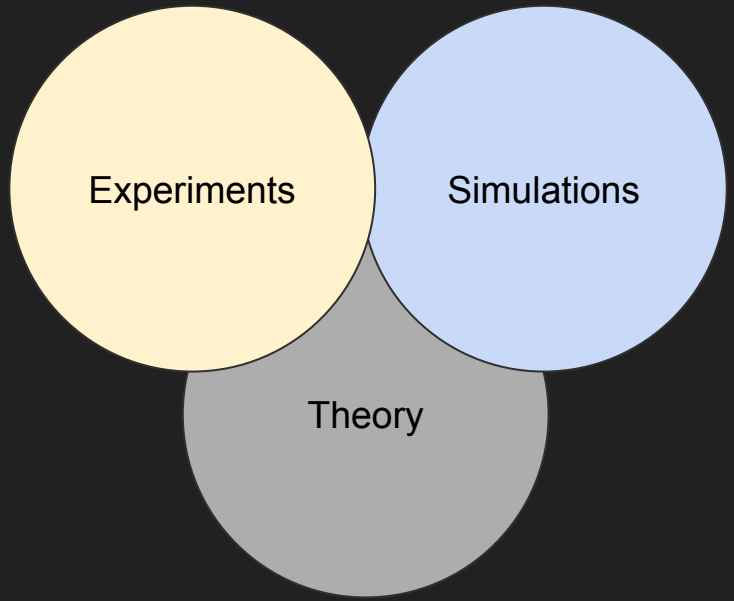
Analysis and Interpretation



Computational physics is multidisciplinary



Theory - Experiments...and Simulations



Why computational physics?

Analytical solutions of physical problems are generally found for systems that are often too simplified.

The vast majority of realistic problems in physics are *complex* and require a numerical approach.

For example:
→ Non-analytical solutions or no mathematical tools able to describe them
→ Too many degrees of freedom: Vlasov equation in 3D/3V + time + electromagnetic fields
→ Chaotic, stochastic, highly-non-linear….
→ Solution is known but too many "particles" and need for fast computations

Numerical methods and algorithms

Depending on the physical problem, many numerical techniques exist for

→ finding roots (i.e. $f(x) = 0$)

→ system of linear equations

→ ordinary and partial differential equations

→ integration,

→ data fitting

→ discrete Fourier transform

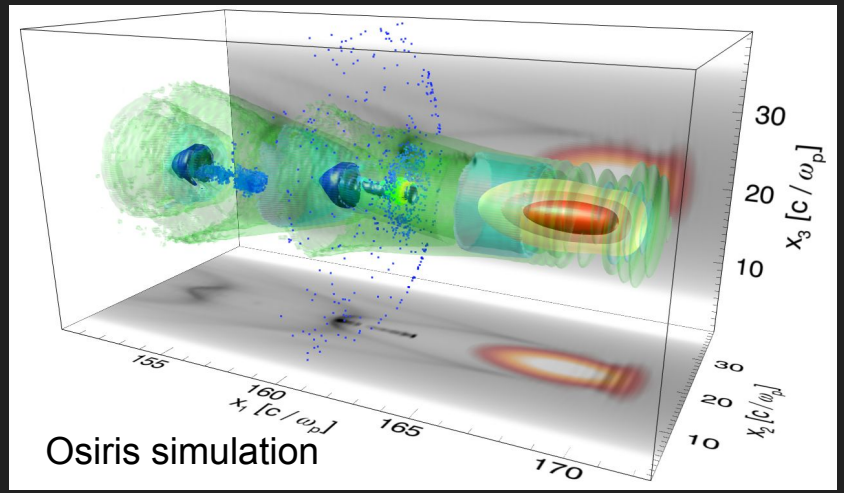
→ matrix eigenvalue

→ ….

Numerical methods and algorithms

"Simple" building blocks are needed to construct complex numerical simulations

For example: Simulation of particle acceleration in laser-plasma interaction
→ interpolation/extrapolation
→ numerical integration
→ ...



Osiris simulation

Simulations can produce x 100 TB of output data that need to be stored and processed
→ big data

Advantages and disadvantages of numerical calculations

- Fast
  - But often Nature is faster...no forecasting (to model 1 ns of laser interaction it takes a day)
- Large memory and data storage
  - Real systems are too big: 12 g of Carbon contain 6×10^{23} particles
- Finite representation of numbers and approximation of equations → errors
  - A good scientist needs to know the limitations of computational results
- ….

The Numerical Methods course

Objectives of the course

1. Learn basic and not-so-basic numerical methods and algorithms
2. Learn to implement different numerical methods and algorithms
3. Use computers to solve problems in physics and interpret the results

...and also:

→ learn PYTHON, jupyter notebooks, ...

→ work with others to solve problems

Python

- high-level, interpreted language
- very flexible, thousands of libraries
- standard choice in many R&D applications and beyond
- huge amounts of information and documentation online

    → *it can be daunting to find the correct/relevant information*

Check out the website: https://www.python.org/
List of python packages: https://pypi.org/

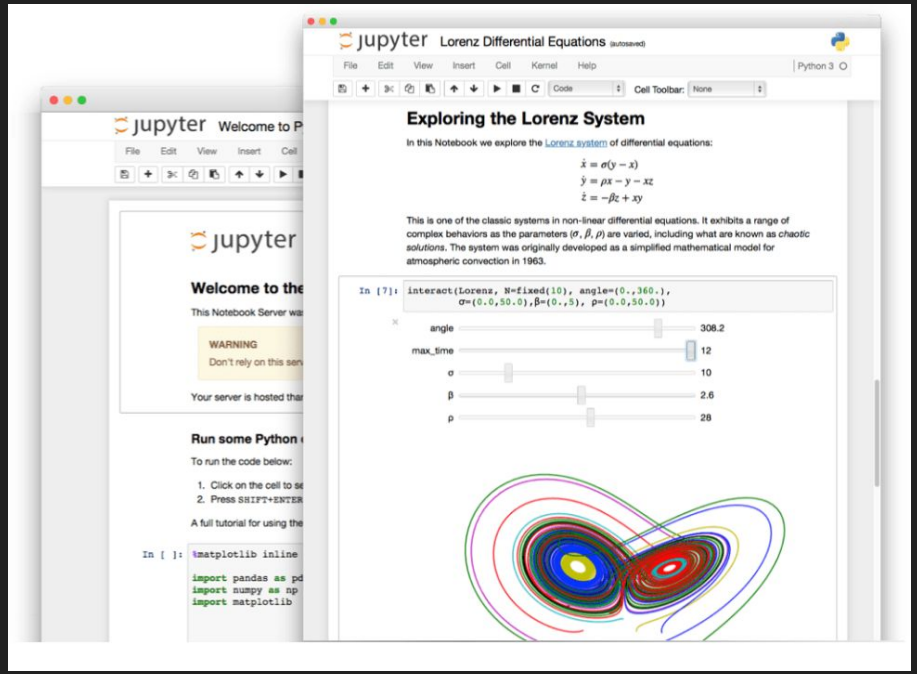
Jupyter notebook

"The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text."

Best way to learn is to try it!!

JupyterLab 2.0: Jupyter's
Next-Generation Notebook Interface

website: https://jupyter.org/



Organization of the course

website: https://sites.google.com/view/paris-physics-master/home

12 lectures of 5 hours
→ Classes are split in ~ 2h theory and ~ 3h practical work
→ Practical work is done on a computer using jupyter notebooks
→ Mixture of in-person and virtual classes
→ Classwork is done individually or in pairs (preferable)
→ work ***must*** be submitted before the following class (i.e Thursday before midnight)

Course note is based on
→ continuous assessment 25% x 2 (50%)
→ classwork notebooks (50%)

Classes are in different rooms and computers are unreliable…
bring a USB key to make a backup (or learn to use GIT)

Course content

Lecture 1 → Introduction and python language - Andrea
Lecture 2 → Roots finding, Interpolation - Andrea
Lecture 3 → Extrapolation and linear regression - Andrea
Lecture 4 → Integration - Andrea
Lecture 5 → Minimization / Optimization 1D - Sebastien
Lecture 6 → Multi-dimensional minimization / optimization - Sebastien
Lecture 7 → Ordinary Differential Equations - part I - Sebastien
Lecture 8 → Ordinary Differential Equations - part II - Sebastien
Lecture 9 → Partial Differential Equations - part I - Sebastien
Lecture 10 → Partial Differential Equations - part II - Sebastien + CC 1
Lecture 11 → PIC methods and Montecarlo methods I - Andrea
Lecture 12 → PIC methods and Montecarlo methods II - Andrea + CC 2

Workflow for the practical part of our classes

1. Download the course material from: https://sites.google.com/view/paris-physics-master/home
2. Open the .ipynb files in jupyter notebook
3. WRITE YOUR NAME IN THE FIRST LINE OF THE NOTEBOOK
4. Do some great work and save
5. Upload the .ipynb file to the same website, where there is a dedicated link to upload

The note for the practical work is global, i.e. you will not get a note for each report.

password: PPM2020!

A little warning…

On the internet (or in this class) you can find
almost all the solutions to *almost* all the problems.

Using other people's work as if it was your own and without proper acknowledgement is wrong!

Plagiarism is never acceptable and it will land you into serious troubles.

If you use other people's work/ideas just say it. There is nothing wrong with that. Yet, you are expected to at least try first.

There is a fine line between copying a "snippet" of code and an entire program...if in doubt, just ask us

The resources of computational physics

Hardware: personal computer

- single CPU multi core
- 16 GB of memory
- TB of disk

Hardware: workstation and servers

- 2 to 4 CPUs many cores + GPU
- hundreds of GB of memory
- x 10 TB of disk

Hardware: small local clusters

- x 10s of CPUs, x10s cores + few GPU
- hundreds of GB of memory
- x 10 TB of disk

Hardware: large clusters

- x 100s of CPUs, few 1000s cores + many GPU
- 1000s of GB of memory
- x 100 TB of disk

Hardware: High-performance super-computers

- x10000s to few million cores
- hundreds xTB of memory
- many PB of disk

They all run Linux-based operating systems

Application Performance	200 PF
Number of Nodes	4,608
Node performance	42 TF
Memory per Node	512 GB DDR4 + 96 GB HBM2
NV memory per Node	1600 GB
Total System Memory	>10 PB DDR4 + HBM2 + Non-volatile
Processors	2 IBM POWER9™ 9,216 CPUs 6 NVIDIA Volta™ 27,648 GPUs
File System	250 PB, 2.5 TB/s, GPFS™
Power Consumption	13 MW
Interconnect	Mellanox EDR 100G InfiniBand
Operating System	Red Hat Enterprise Linux (RHEL) version 7.4


Barcelona supercomputing centre

Software: languages, libraries and visualization

- languages:
  - C++, Fortran, Python, ...
- softwares:
  - MAPLE, MATHEMATICA, ...
- libraries:
  - many libraries exist to perform numerical calculations (e.g. linear algebra, fitting, etc.)
  - MPI for parallel computing, openMP, ...
  - In this course we will make extensive use of numpy
- visualization
  - plotting, 2D and 3D visualization
  - VisIt, Paraview, Mayavi, ….
- version control and other utilities to keep track of your work and collaborate
  - git, gitkraken, ….

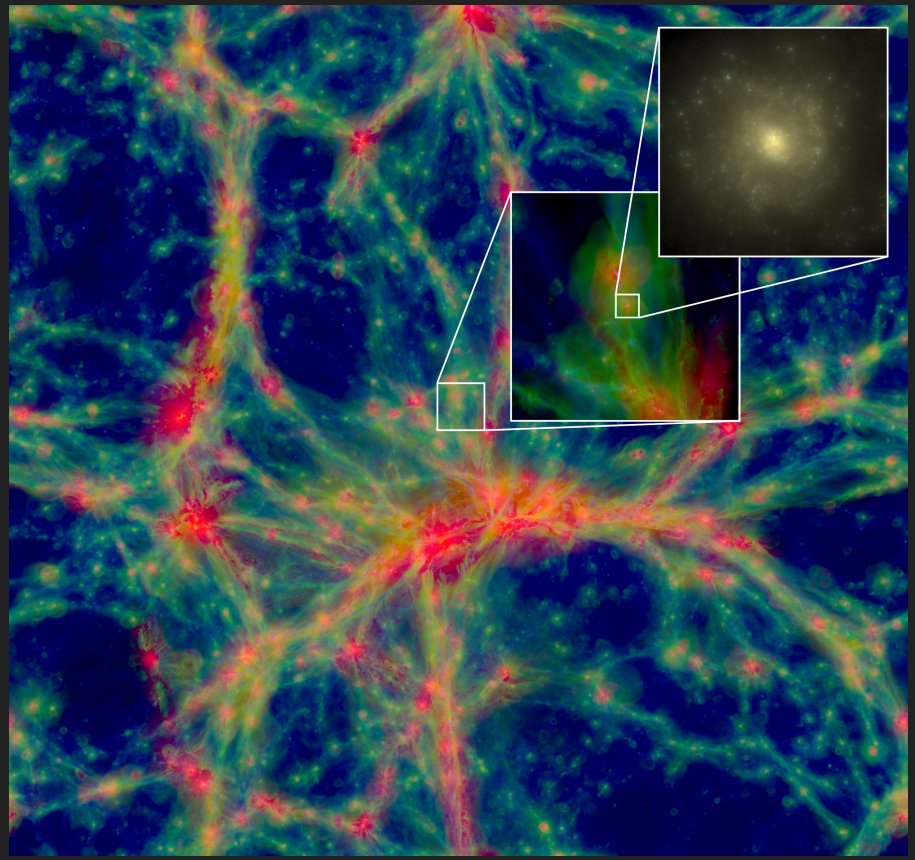
Examples of numerical simulations

Cosmological simulations

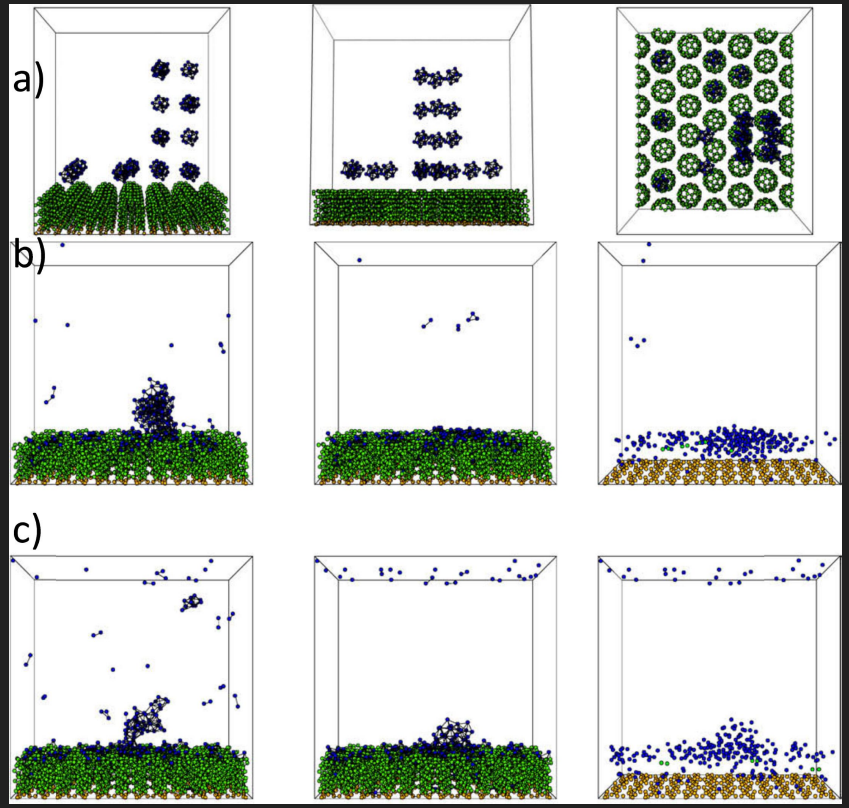
The EAGLE simulation is one of the largest cosmological hydrodynamical simulations ever, using nearly 7 billion particles to model the physics.

It took more than one and a half months of computer time on 4000 compute cores of the DiRAC-2 supercomputer in Durham.

It was performed with a heavily modified version of the public GADGET-2 simulation code.

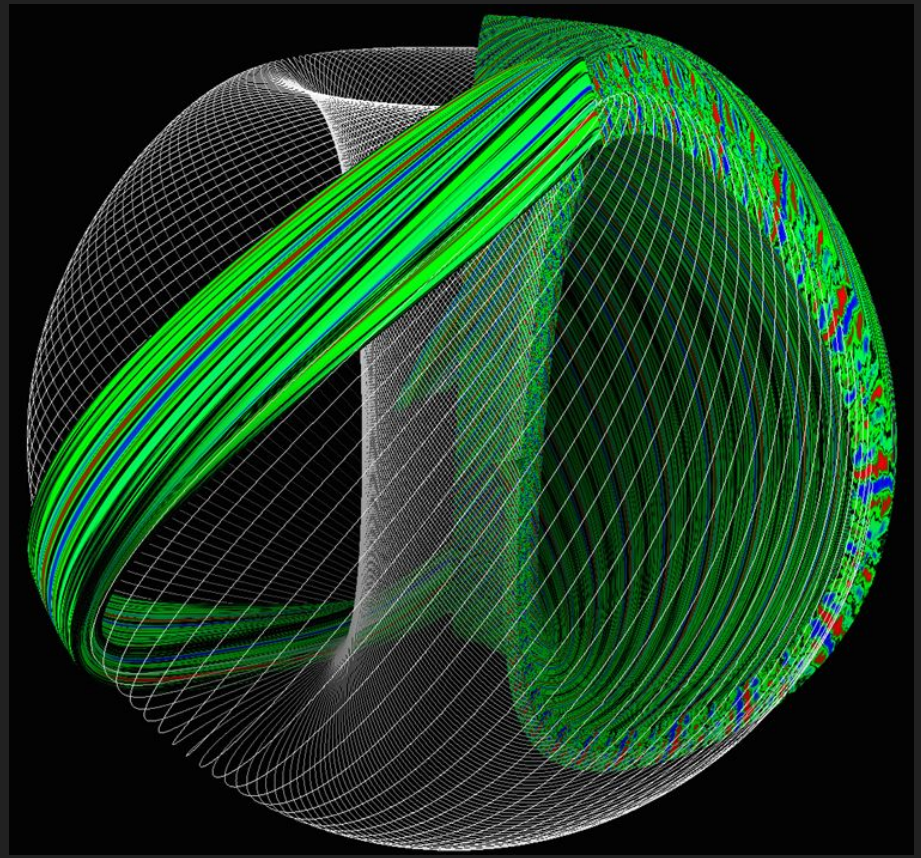


Vapor deposition of Ni and Au clusters on vertically aligned carbon nanotubes.



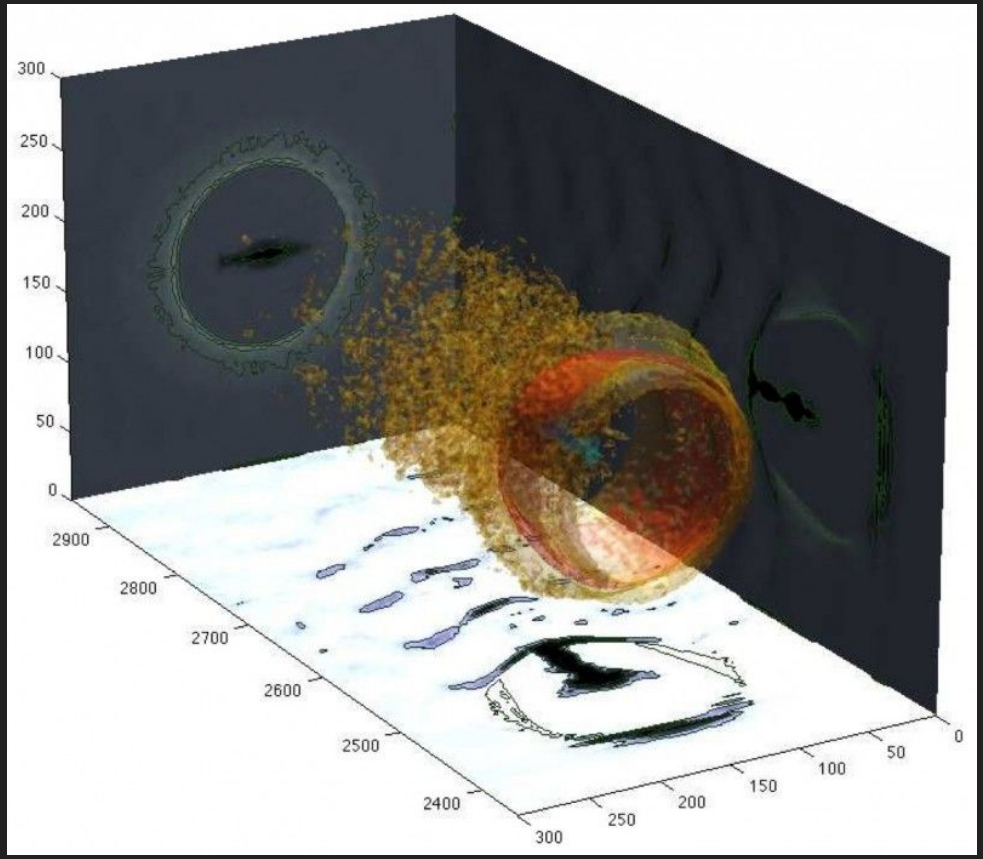
Molecular dynamics simulations of supported metal nanocatalyst formation by plasma sputtering
Pascal Brault and Erik C.Neyts

5-D gyrokinetic turbulence simulations in a tokamak



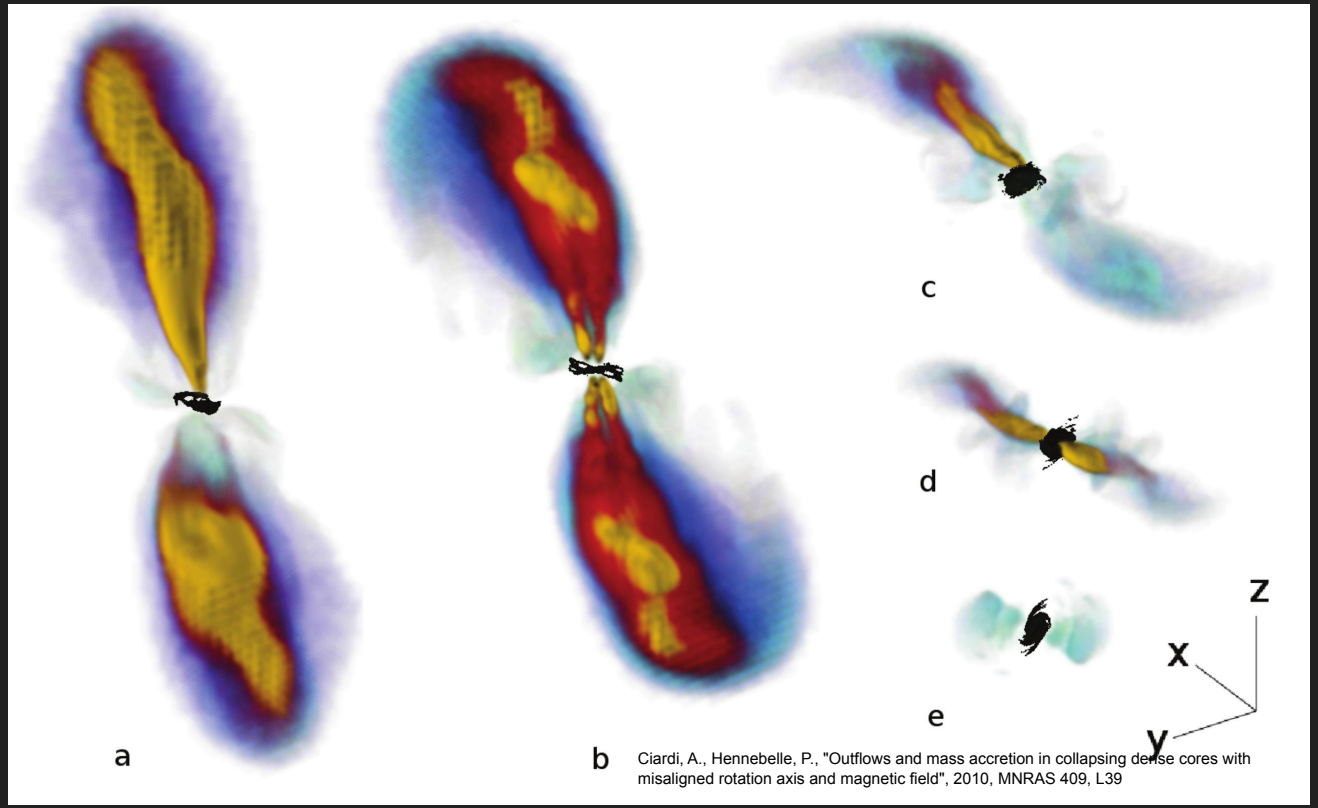
https://w3.pppl.gov/~hammett/viz/viz.html

Ultra high intensity laser-plasma interaction



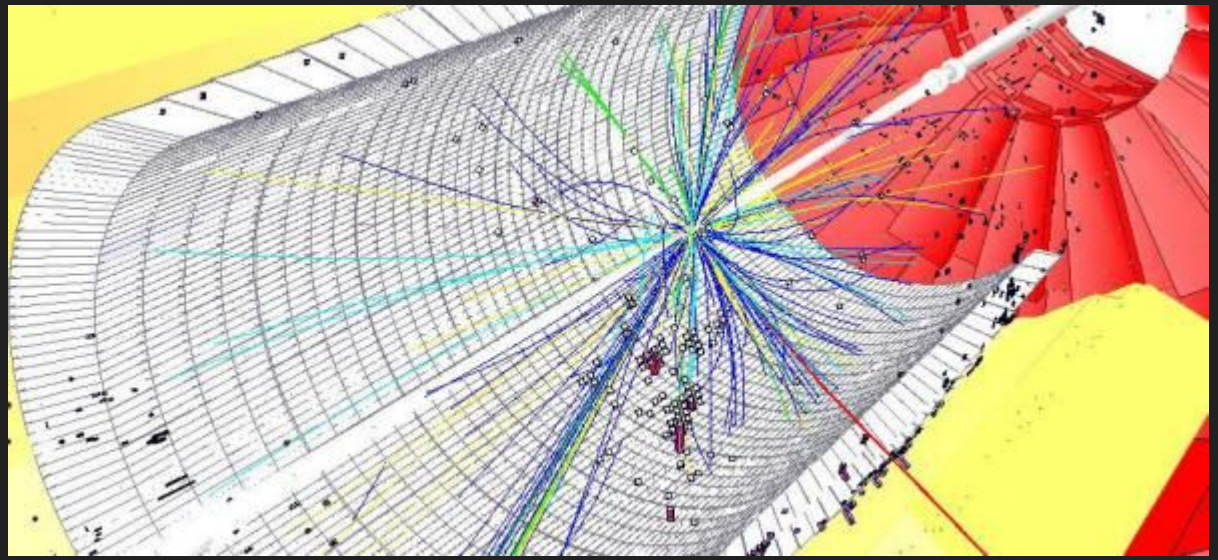
https://cuos.engin.umich.edu/researchgroups/hfs/research/theory-and-computation/
The bubble formed in the wake of the HERCULES pulse, according to a 3D Particle-In-Cell simulation using the OSIRIS 2.0 framework

Magnetohydrodynamic simulations of star formation



Ciardi, A., Hennebelle, P., "Outflows and mass accretion in collapsing dense cores with misaligned rotation axis and magnetic field", 2010, MNRAS 409, L39

Simulations of high-energy particle collisions



http://lhcathome.web.cern.ch/projects/test4theory/high-energy-physics-simulations