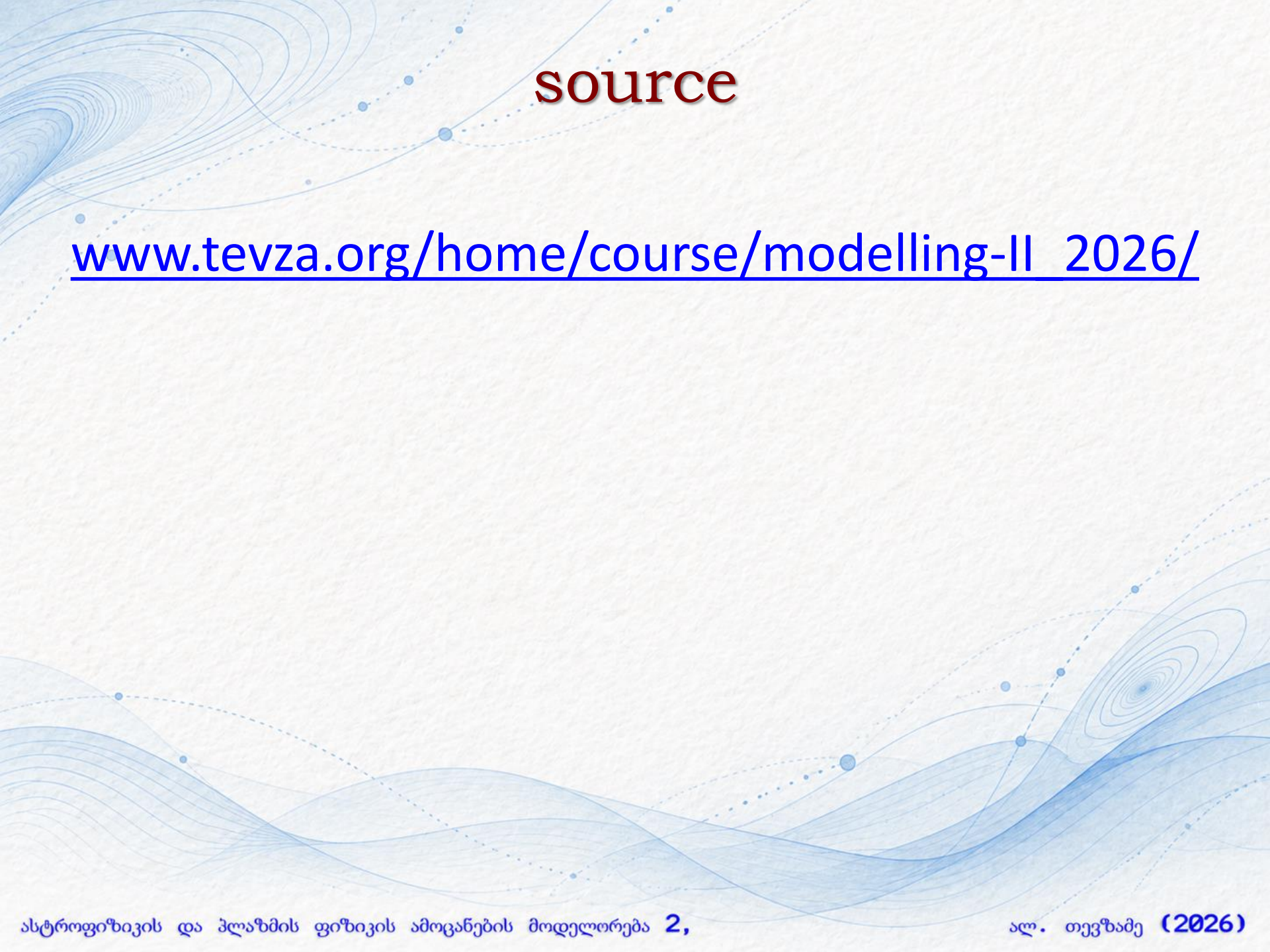




ასტროფიზიკის და პლაზმის ფიზიკის ამოცანების მოდელირება - 2

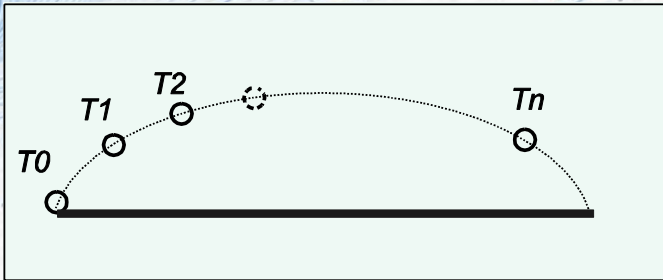
ლექცია 1



source

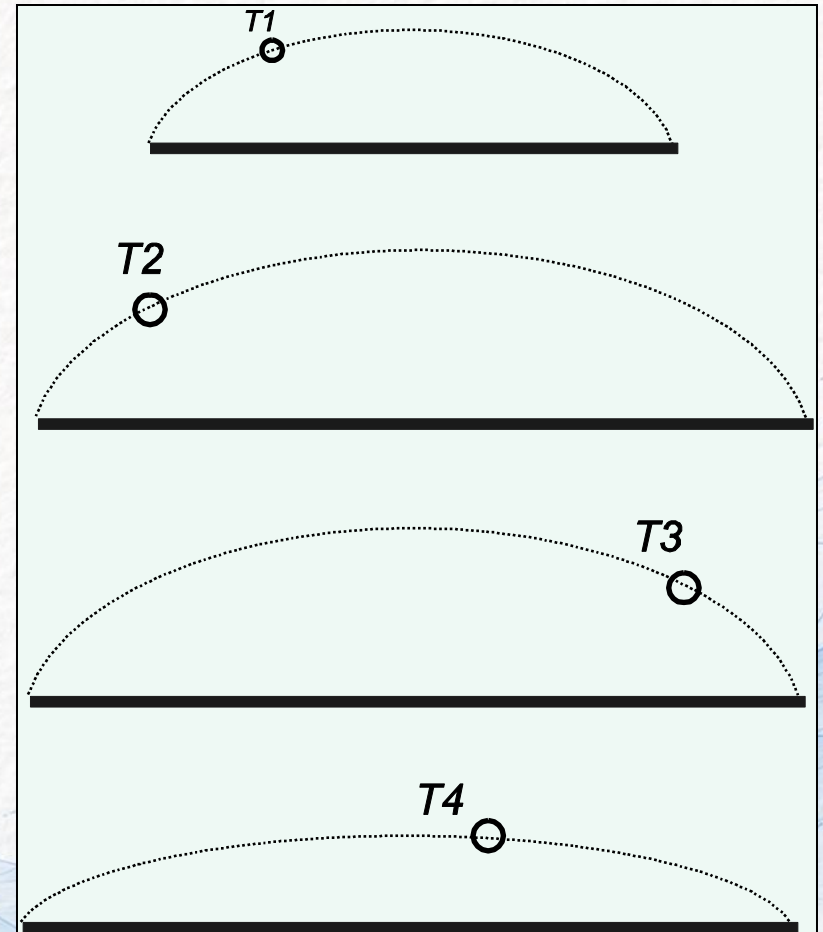
www.tevza.org/home/course/modelling-II_2026/

Why model?



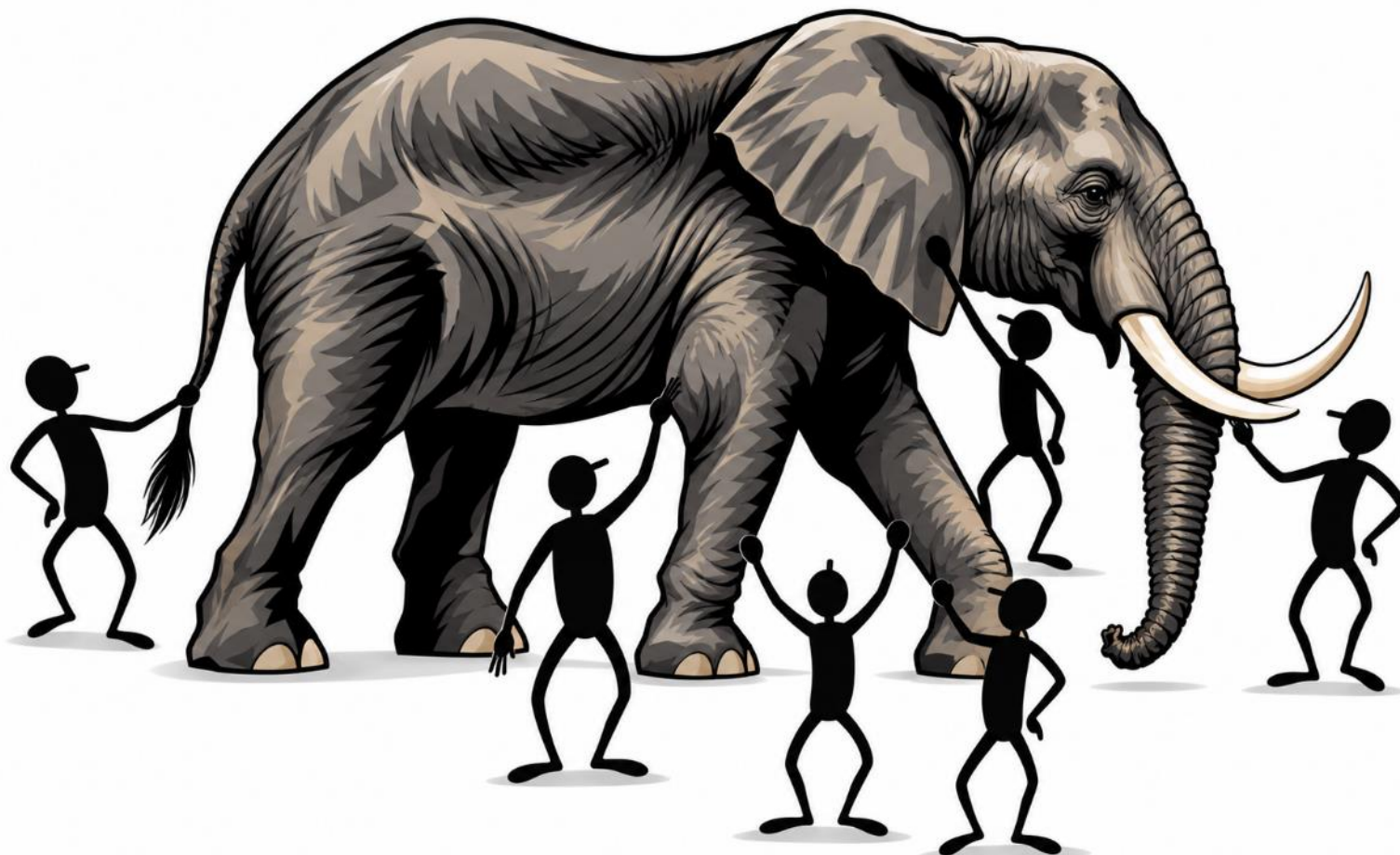
Consequent data

Inconsequent data:



Modelling Elephant

How good is the model?

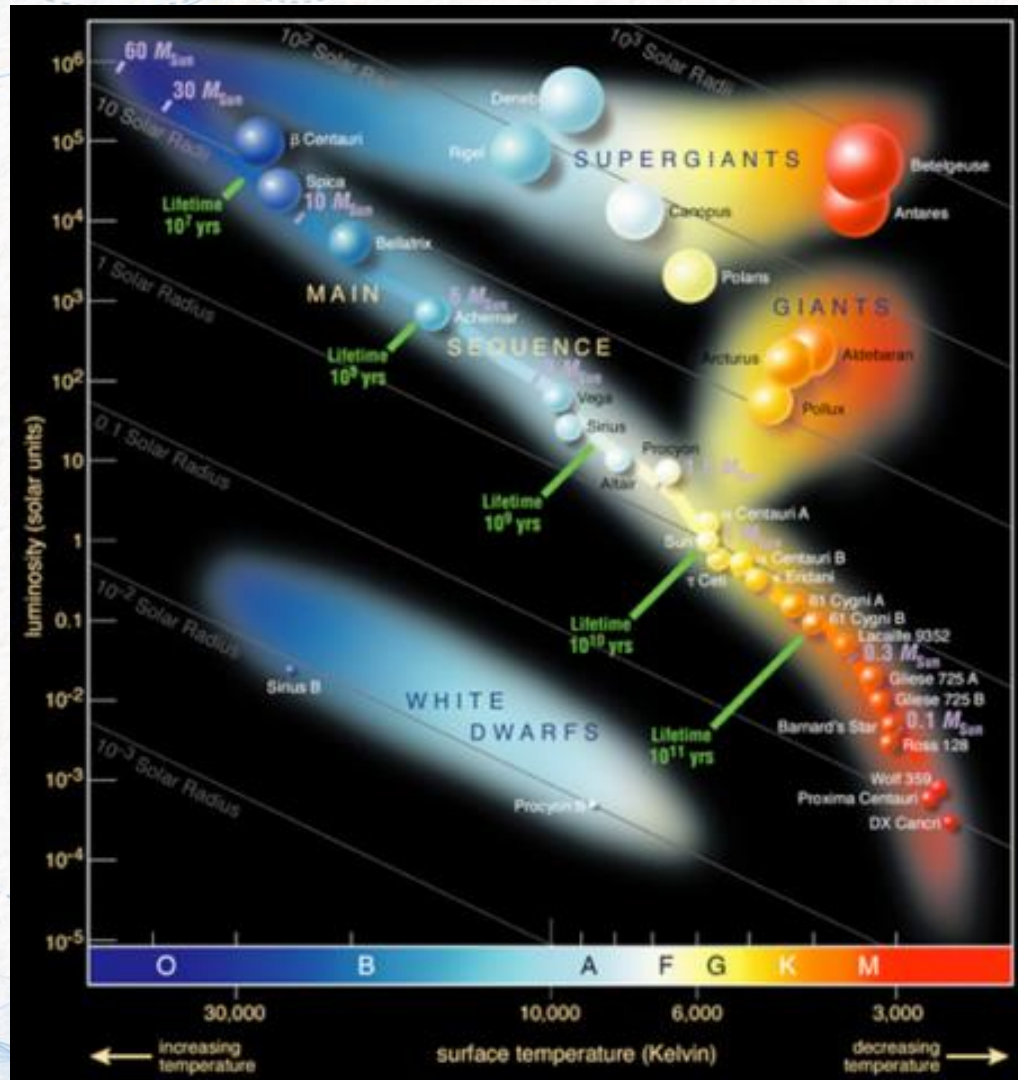


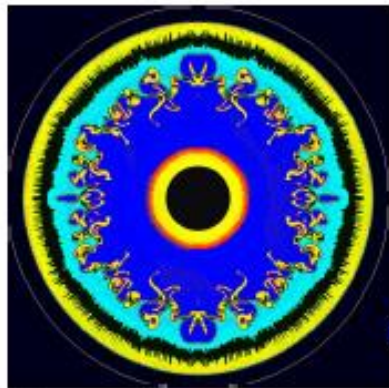
Astrophysical Simulations

Different physics:

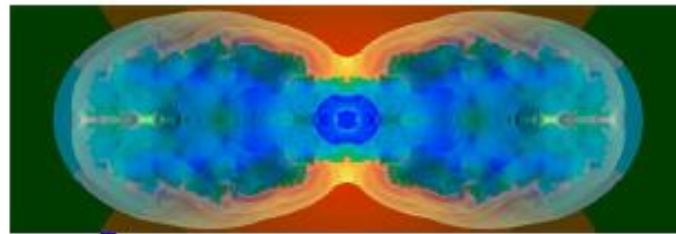
- Different classes
- Different stages of the evolution
- Different scales

Hertzsprung-Russell Diagram

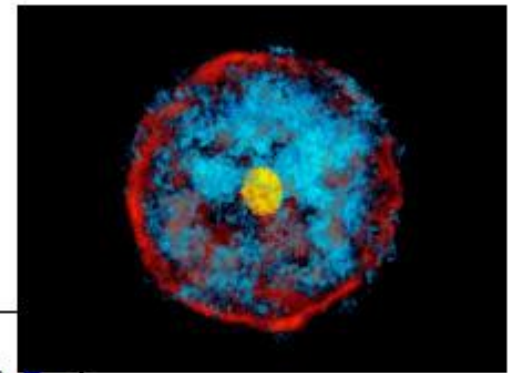




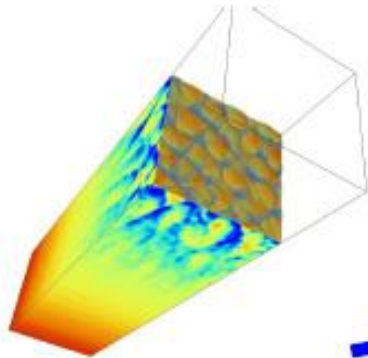
Core-collapse supernovae



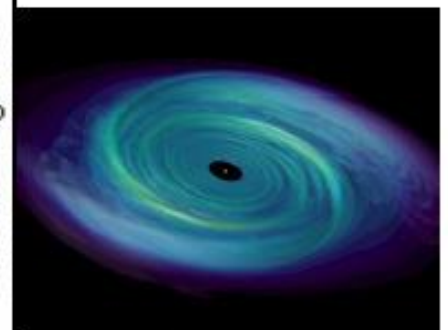
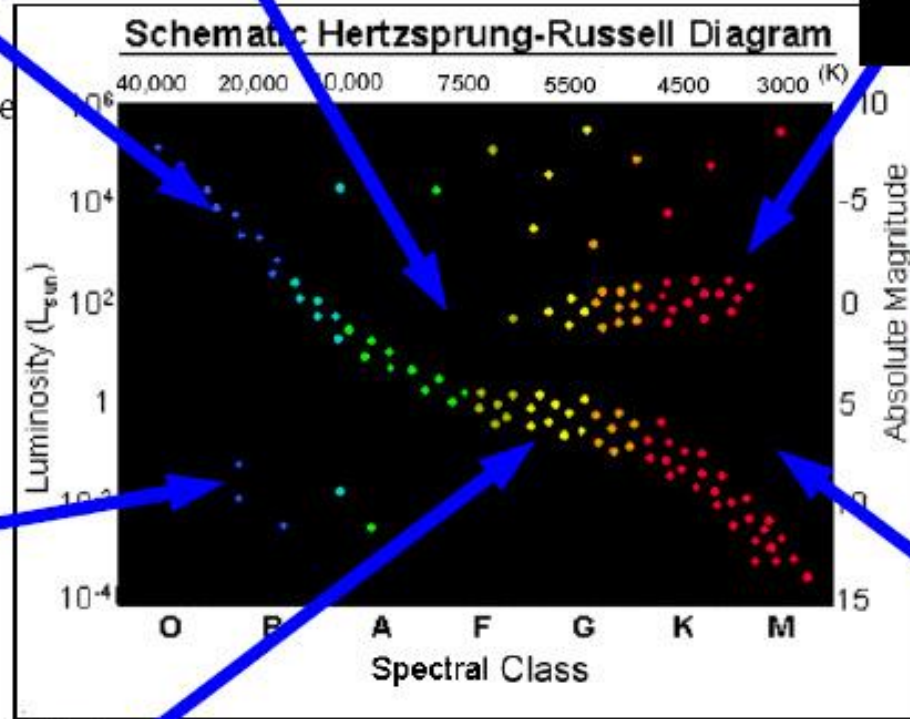
Planetary nebulae



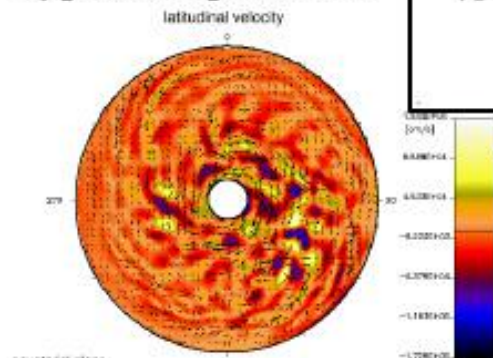
Red giant evolution



Type Ia supernovae

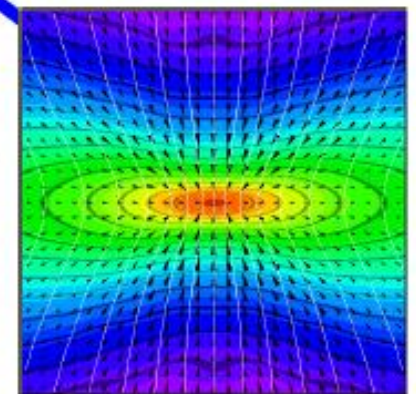


Binary stars

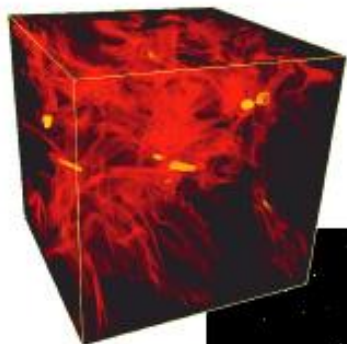


Main-sequence evolution

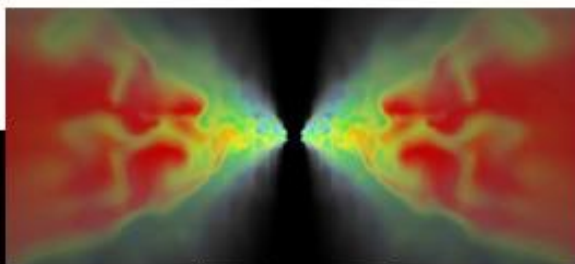
Computational Stellar
Evolution



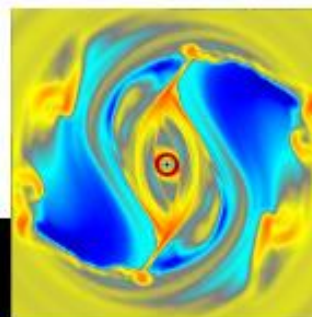
Star formation



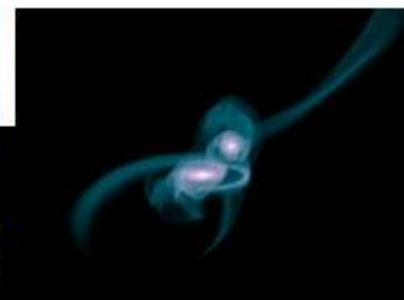
Molecular clouds



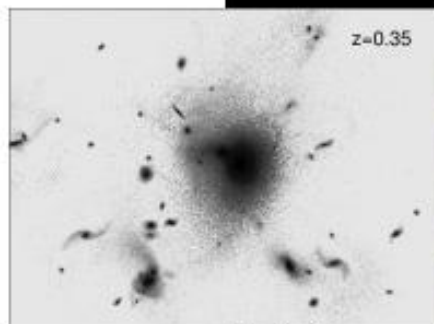
Supermassive black holes



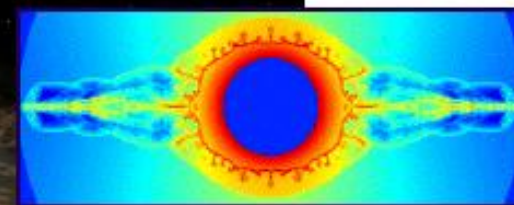
Evolution of disk structure



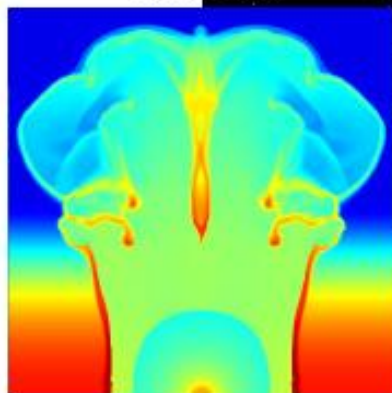
Galaxy mergers



Galaxy evolution in clusters

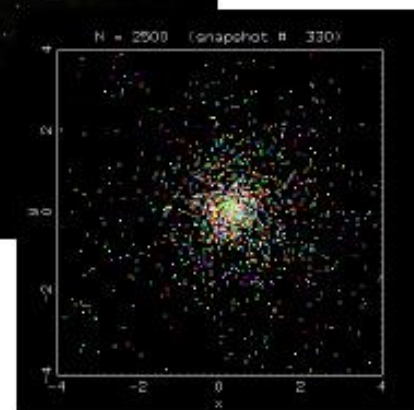


Supernovae and the interstellar medium



Superbubbles

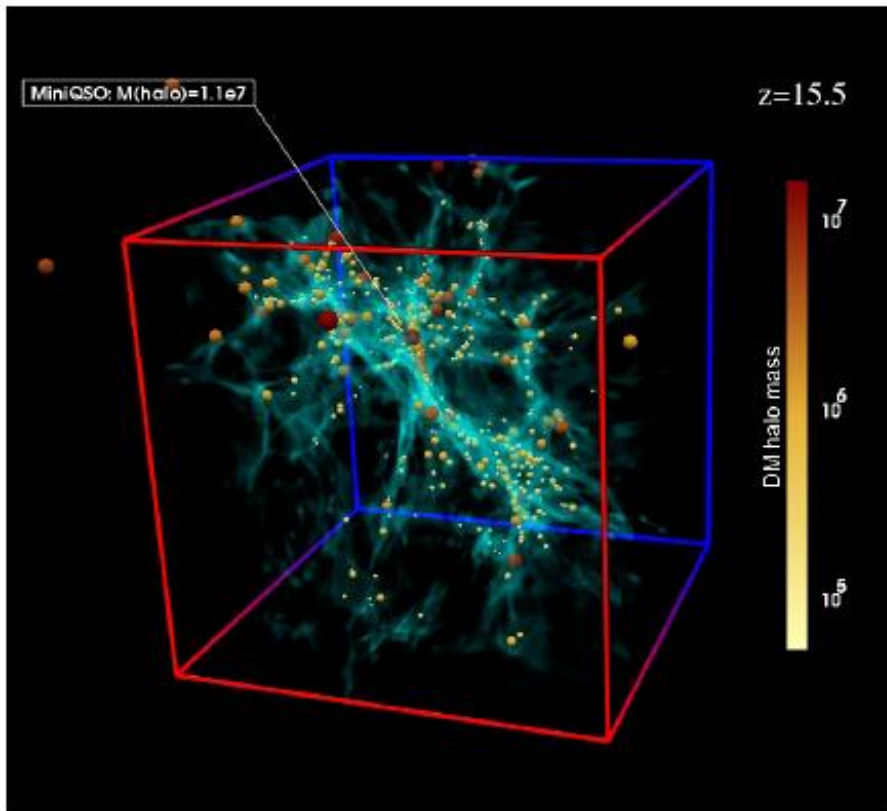
Computational Galactic Evolution



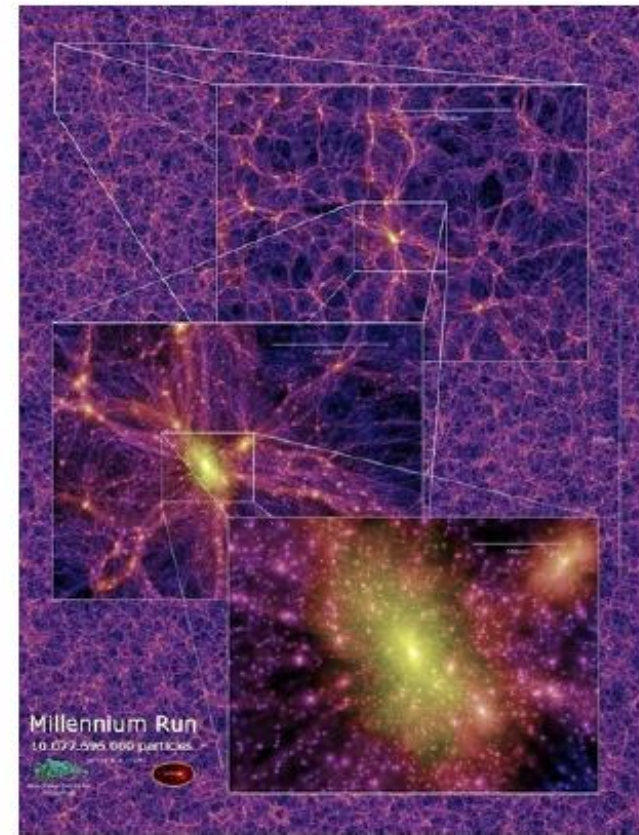
Star clusters

Computation on the Cosmological Scale...

- On the scale of the Universe, the cosmological scale factor is evolved.
- Self-gravity dominates the evolution



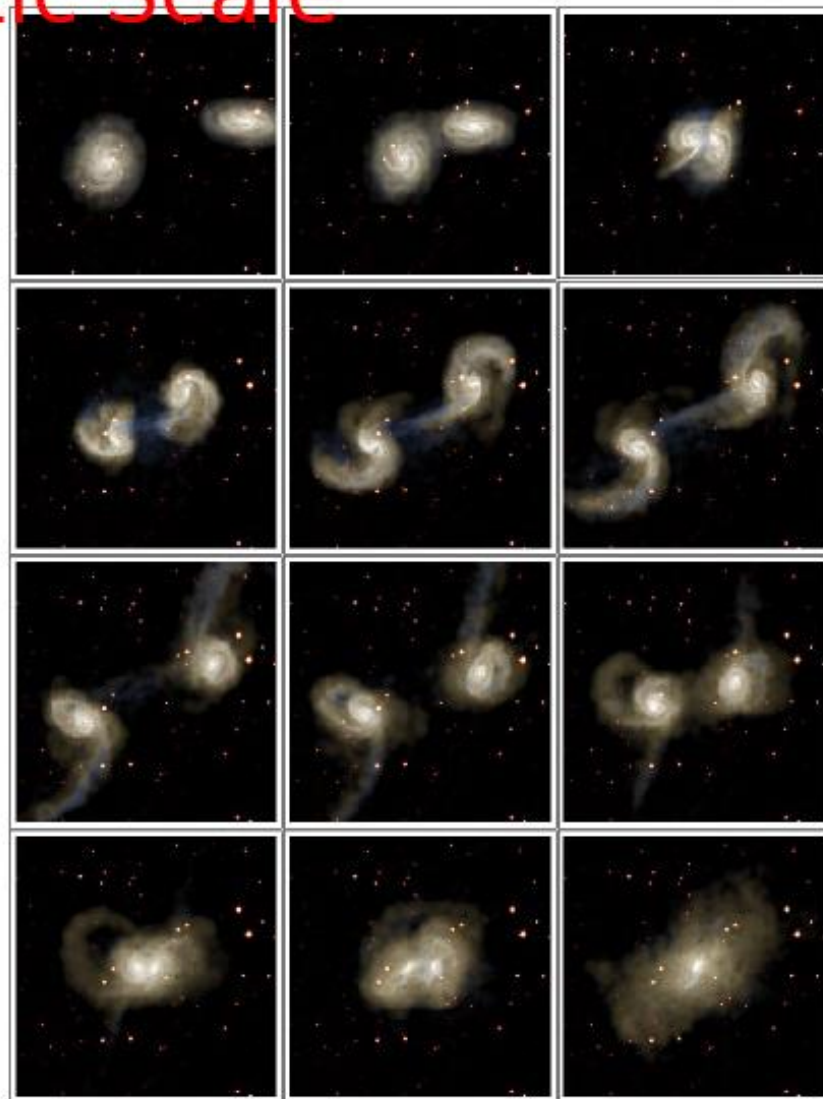
The first mini quasar effects on the surrounding IGM (Kuhlen et al.)



Simulating the growth of structure and the formation of galaxies. (Springel et al. 2005)

...the Galactic Scale

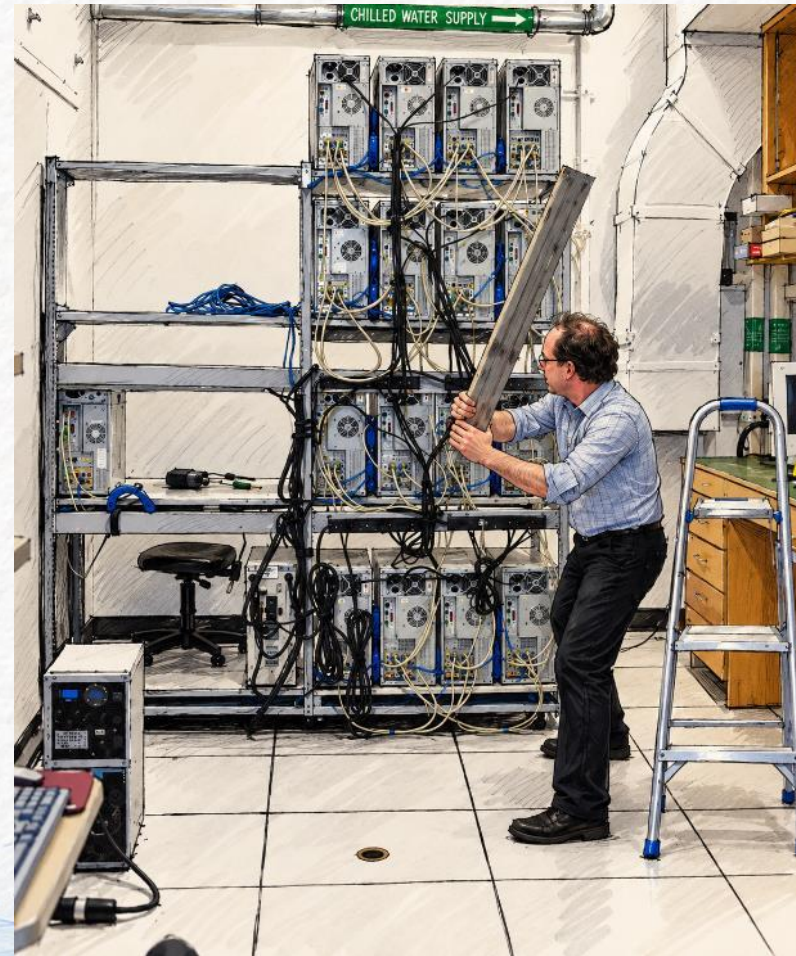
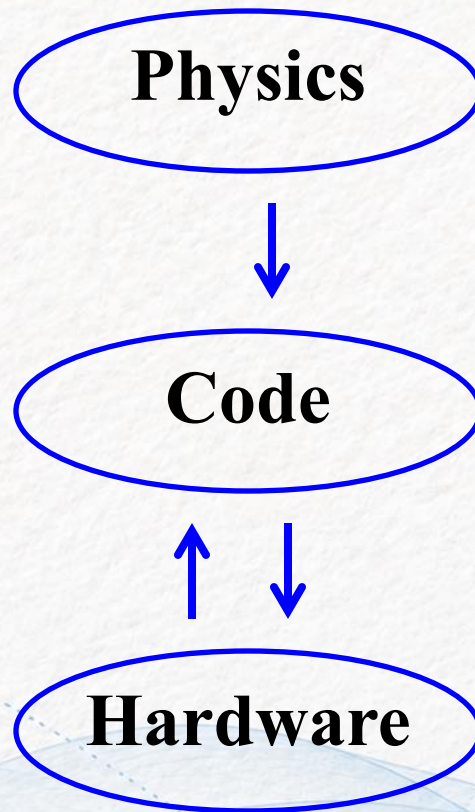
We can understand how galaxies interact and merge. It takes 100s of million years to play out in nature—we can see the evolution at a much accelerated pace.



Colliding and merging galaxies.
Springel & White (1999)

Numerical Modelling

Major aspects:



Publicly available simulation codes — 2026

Code	Numerical approach	Main physics	Parallel / accelerator
Athena++	Godunov finite-volume, AMR	HD, MHD, radiation, GRMHD	MPI, HPC
PLUTO / gPLUTO	Finite volume / finite difference	HD, MHD, relativistic HD/MHD	MPI; GPU in gPLUTO
FLASH	AMR, finite volume	HD, MHD, gravity, nuclear physics	MPI, HPC
RAMSES	AMR	HD, MHD, gravity, radiation, cosmology	MPI, HPC
Enzo	AMR + particles	HD/MHD, gravity, cosmology, chemistry	MPI, HPC
AREPO	Moving Voronoi mesh + TreePM	HD/MHD, gravity, cosmology	massively parallel
GADGET-4	N-body + SPH, TreePM/FMM	gravity, cosmology, gas	MPI, HPC
GIZMO	Mesh-free MFM/MFV/SPH	HD/MHD, gravity, galaxy formation	MPI, HPC
Smilei	Particle-in-cell	kinetic / relativistic plasma, laser plasma, astrophysical plasma	MPI + OpenMP, HPC
Einstein Toolkit	finite difference + AMR	numerical relativity, GR hydrodynamics	MPI, HPC

PLUTO

<http://plutocode.ph.unito.it/> A Riemann solver for HD/MHD/RMHD with AMR. Parallel. C/C++

SNOOPY

<http://ipag.osug.fr/~lesurg/snoopy.html> Spectral, incompressible MHD, parallel

PENCIL

<https://github.com/pencil-code/pencil-code/> MHD Cartesian. A higher order non-conservative advection method. Turbulence. Parallel. FORTRAN

FLASH

<https://flash.rochester.edu/> Cartesian HD, modules, AMR, parallel.

ZEUS

<http://www.astro.princeton.edu/~jstone/zeus.html> (M)HD. Staggered grid, Cartesian, cylindrical, polar, gravity, self-gravity and radiation transfer.

ATHENA++

<https://www.athena-astro.app/> Riemann solvers (including also Roe's algorithm). Cartesian. HD/MHD, GRMHD, AMR, parallelization (MPI) etc.

GADGET

<http://www.mpa-garching.mpg.de/gadget/> SPH and N-body code for astrophysics.

Modern Parallel Computer Architectures

1. Shared-Memory Multiprocessor (Multicore / NUMA)



- Multiple CPU cores share a common address space
- Memory may be uniform (UMA) or non-uniform (NUMA)
- Threads communicate through shared variables
- Typical programming: OpenMP, pthreads
- Example: modern multi-socket CPU server / workstation

2. Distributed-Memory Cluster



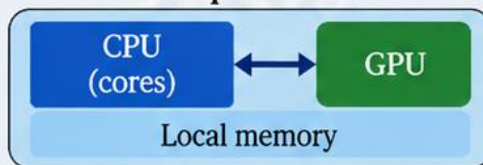
- Each compute node has its own processors and local memory
- Nodes communicate through a high-speed network
- Data must be explicitly exchanged between nodes
- Typical programming: MPI
- Example: HPC cluster / supercomputer

3. Heterogeneous CPU-GPU Systems



- CPUs coordinate computation; GPUs execute massively parallel kernels
- GPUs contain thousands of lightweight processing cores
- Very high throughput for suitable numerical algorithms
- Typical programming: CUDA, HIP, OpenACC, OpenMP offload
- Example: modern GPU-accelerated supercomputer

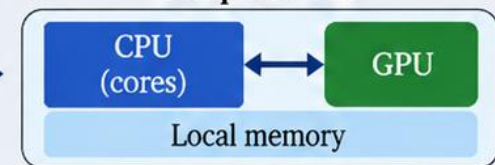
Compute node



High-speed interconnect

(InfiniBand, Slingshot, etc.)

Compute node



Modern HPC usually combines all three levels:

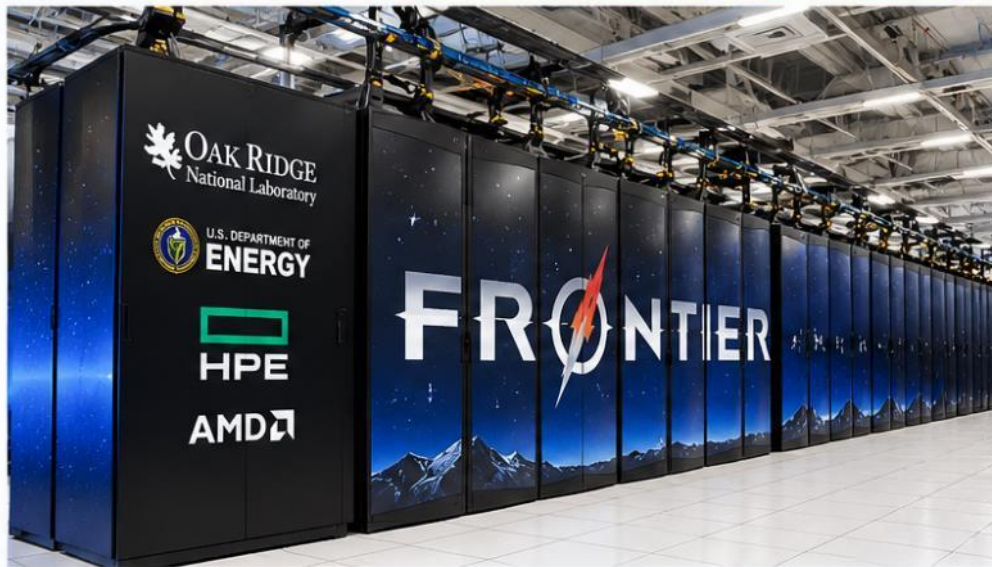
MPI between nodes + OpenMP within CPU nodes + GPU acceleration

Top500 Supercomputers • June 2026 (top500.org/lists)

#	Site (Country)		Supercomputer (vendor)	Processors / Accelerators	Total cores (approx.)	Power (approx., kW)
1		Oak Ridge National Laboratory (USA) 	Frontier (HPE)	AMD EPYC 64C + AMD Instinct MI250X	~ 8.7 million (CPU + GPU)	21,100 kW
2		Jülich Forschungszentrum (Germany) 	JUPITER (Atos)	SiPearl Rhea1 (72C) + NVIDIA GH200	~ 7.1 million (CPU + GPU)	17,000 kW
3		Lawrence Livermore National Laboratory (USA) 	El Capitan (HPE)	AMD EPYC 64C + AMD Instinct MI300A	~ 11.0 million (CPU + GPU)	29,581 kW
4		Argonne National Laboratory (USA) 	Aurora (HPE)	Intel Xeon Max (52C) + Intel Ponte Vecchio	~ 9.2 million (CPU + GPU)	38,698 kW
5		RIKEN (Japan) 	Fugaku (Fujitsu)	Fujitsu A64FX (48C)	~ 7.6 million (CPU)	29,899 kW
6		National Supercomputing Center (China) 	Sunway TaihuLight (NRCPC)	Sunway SW26010 (260C)	~ 10.6 million (CPU)	15,371 kW
7		National Supercomputing Center (China) 	Tianhe-2A (NUDT)	Intel Xeon (12C) + Matrix-2000 (57C)	~ 4.9 million (CPU + accel.)	17,808 kW
8		Los Alamos National Laboratory (USA) 	Crossroads (HPE/Cray)	AMD EPYC 64C + AMD Instinct MI250X	~ 3.4 million (CPU + GPU)	7,360 kW
9		Bristol Centre for Supercomputing (UK) 	Isambard-AI (HPE)	NVIDIA Grace CPU + NVIDIA Grace Hopper	~ 2.2 million (CPU + GPU)	5,641 kW
10		CINECA (Italy) 	Leonardo (Atos)	Intel Xeon (64C) + NVIDIA A100 (80GB)	~ 3.5 million (CPU + GPU)	7,404 kW

FRONTIER — Oak Ridge National Laboratory (ORNL)

HPE Cray EX • Oak Ridge Leadership Computing Facility • USA



Key Information (TOP500 June 2026)

TOP500 rank	#3
Rmax (Linpack)	1,353 PFLOP/s (1.353 EFLOP/s)
Total cores (reported)	9,066,176
System power (reported)	24,607 kW (24.6 MW)
System	HPE Cray EX
Location	Oak Ridge National Laboratory (ORNL), Oak Ridge, USA
Facility	Oak Ridge Leadership Computing Facility (OLCF)
Configuration	77 HPE Olympus cabinets, 9,856 AMD compute nodes
Cooling	Liquid-cooled system

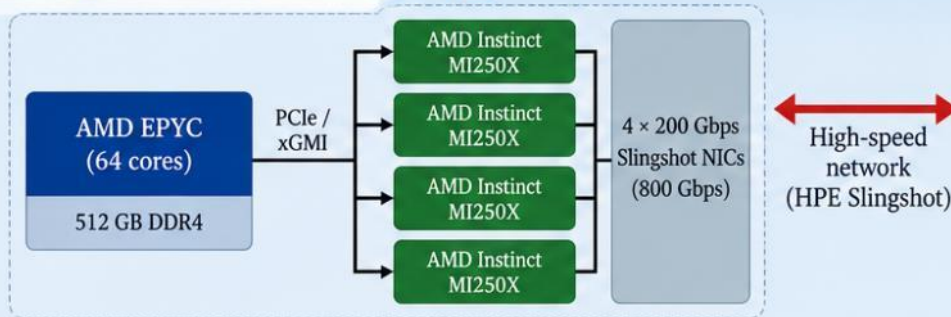
Compute Node Specifications

CPU	1 × 64-core AMD Optimized 3rd Gen EPYC
GPUs	4 × AMD Instinct MI250X
Memory per node	512 GB DDR4 (CPU memory) + 512 GB HBM2e (total GPU memory)
Interconnect	4 × HPE Slingshot 200 Gbps NICs per node (800 Gbps aggregate node injection bandwidth)

Storage

File system	Orion Lustre filesystem, 679 PB usable namespace
-------------	--

Compute Node (HPE Cray EX)



Architecture: CPU + GPU heterogeneous computing • MPI between nodes • GPU acceleration within nodes

Sources: TOP500, June 2026; Oak Ridge Leadership Computing Facility (OLCF).

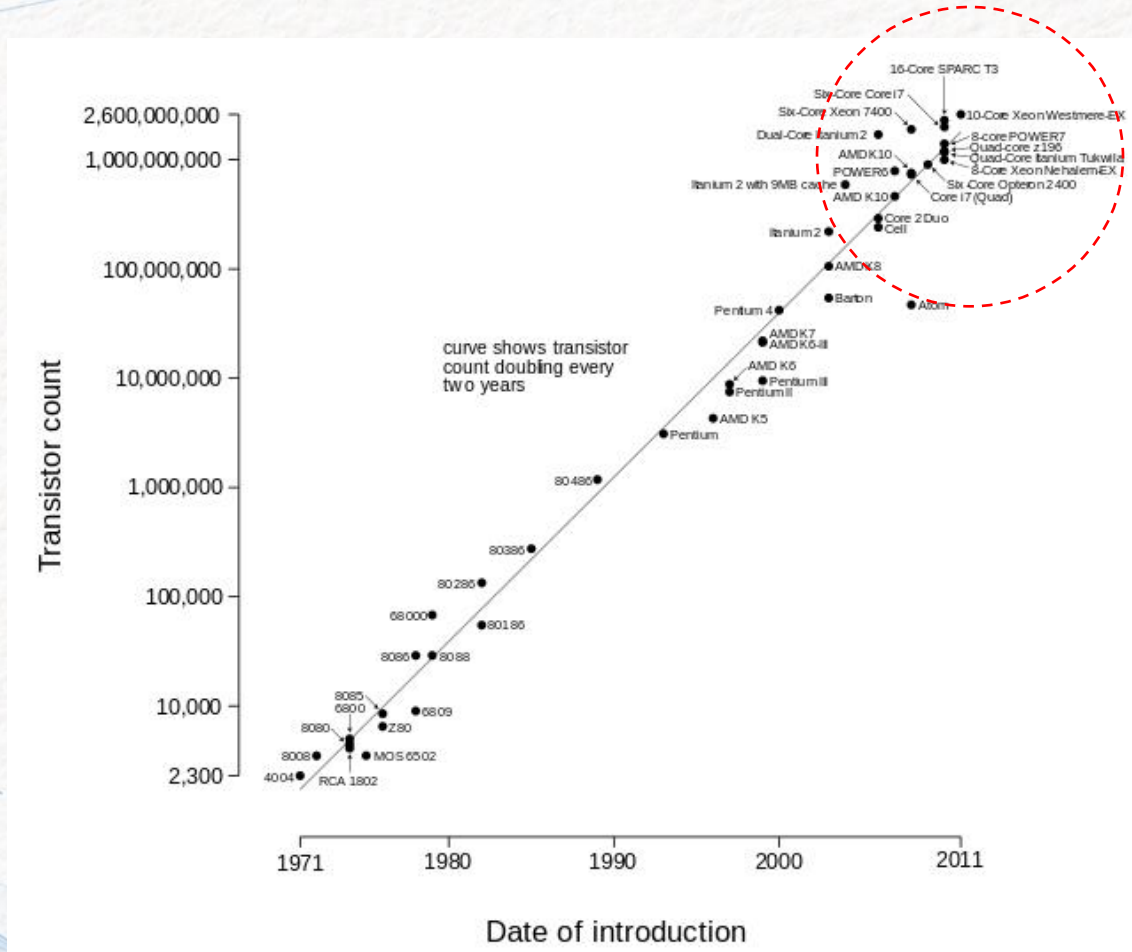
Beowulf Cluster



Perspective

Moor's law

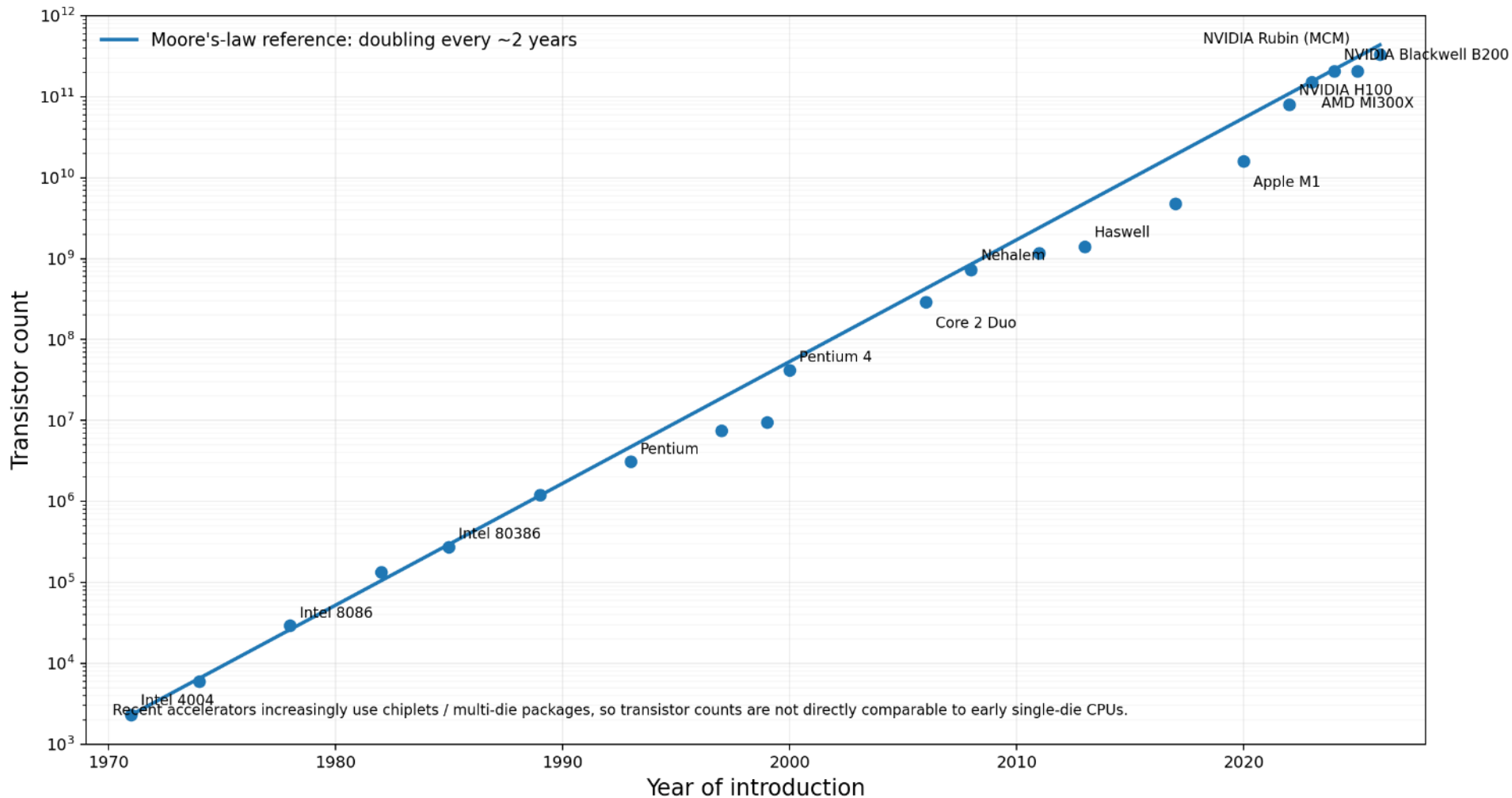
Doubling of the number of transistors on integrated circuits (1.5-2 years)



2016 – End of the Moors roadmap?

Moore's Law

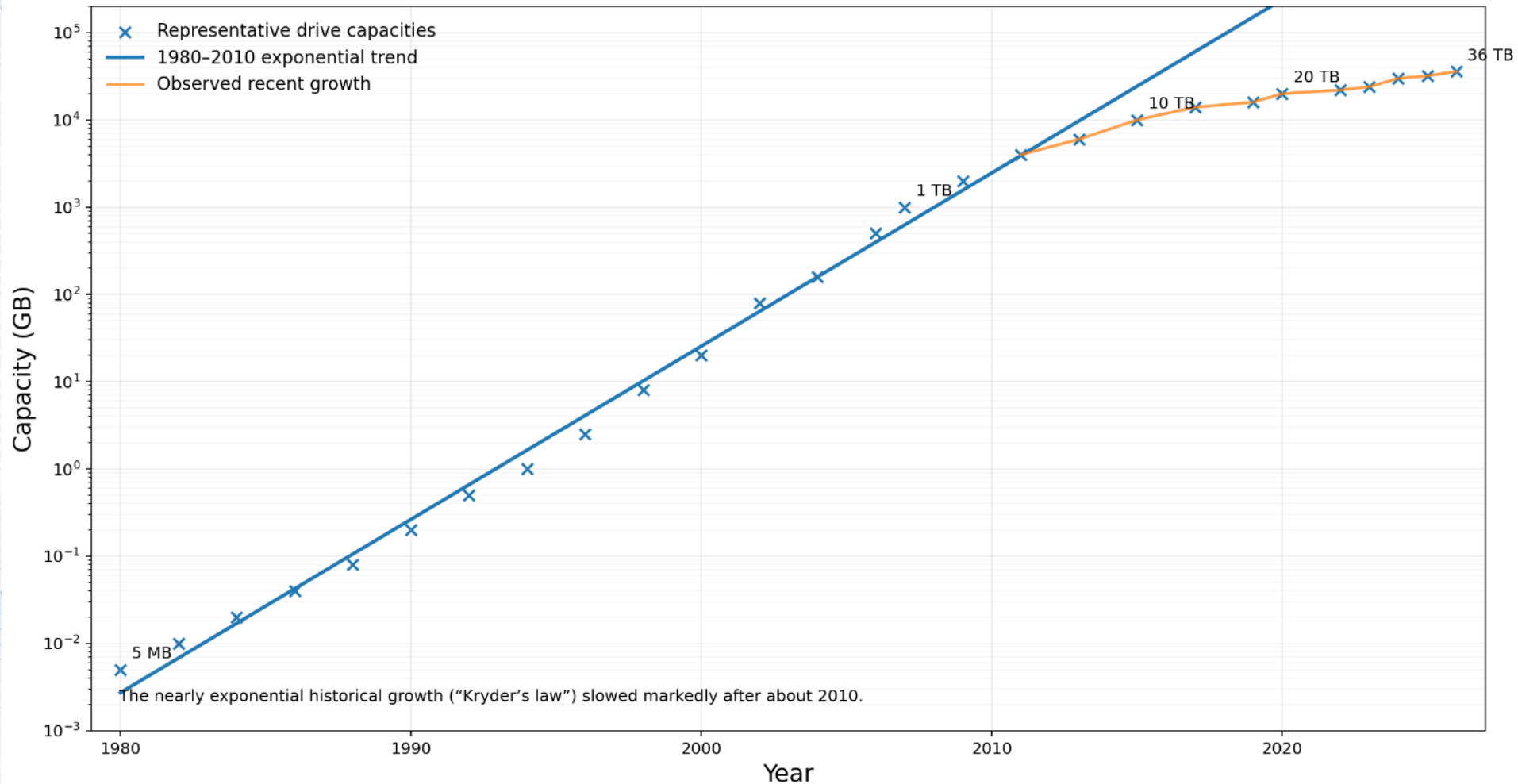
Moore's Law: Representative Processor Transistor Counts, 1971-2026



Kryder's Law

Doubling of the magnetic disk areal storage density (1 year)

Growth of Commercial Disk Capacity, 1980-2026



BASIC CONCEPTS

Method

- Continuum -> Discrete
- Physical Model -> Numerical Model

$(V_x, V_y, V_z, \text{Rho}, P)$

(P_x, P_y, P_z, E, M)

Where to introduce numerical errors:

Discretization, Subgrid, interpolation, etc.

Error propagation science

Workflow

Configuration

- Initial conditions
- Boundary conditions

Calculus

- Compilation
- Run

Data analysis

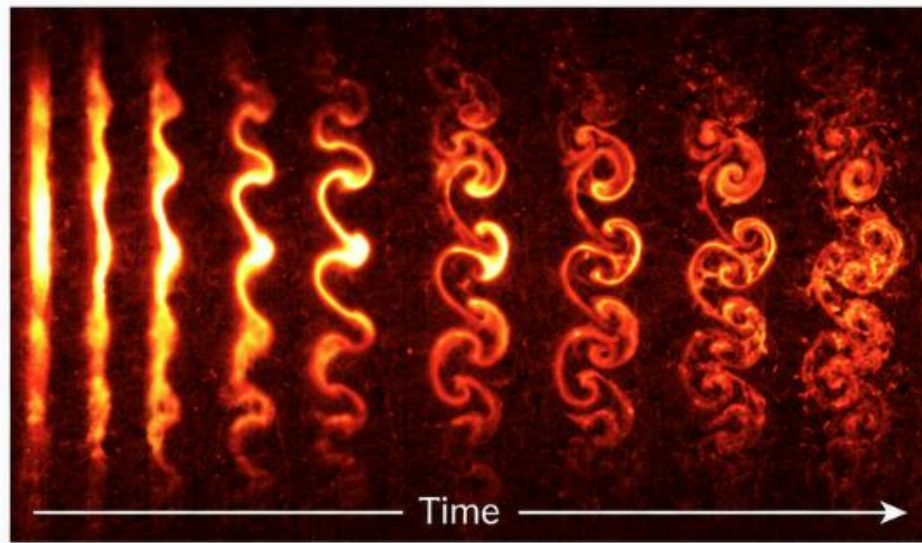
- Post processing
- Visualization

Are my results correct?

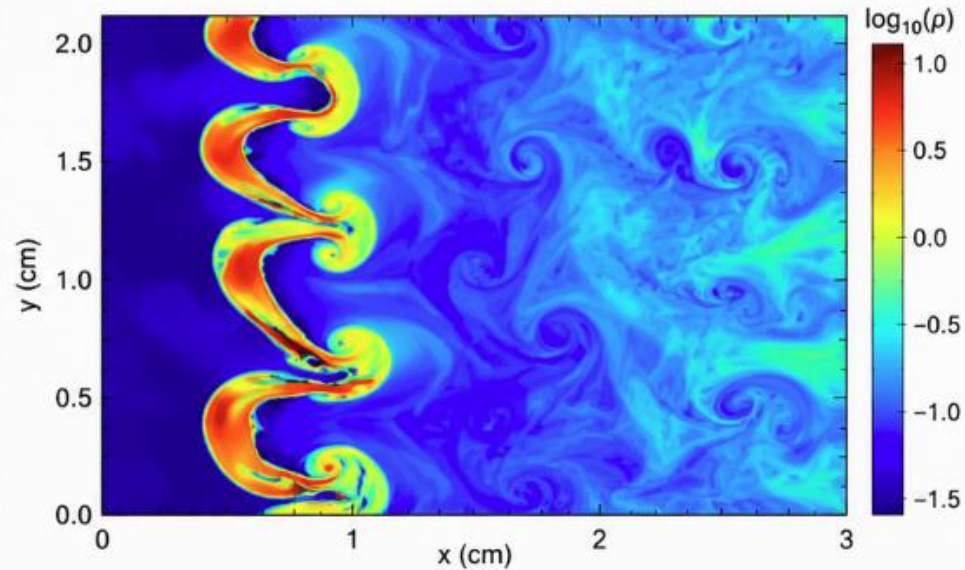
Indicators:

- Resolution Study
- Exact Analytic Solutions
- Different Numerical Methods
- Code Validation

Code Validation



Experiment
(FLASH Center)



Simulation
(LLNL)

Catastrophic cancellation

Calculate the equation with **$a=77617$** and **$b=33096$**

$$y = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}$$

answer depends of the compiler

(C, Fortran, Matlab) processor type!

$$\stackrel{?}{=} 5.76461 \dots \times 10^{17}$$

$$\stackrel{?}{=} 6.33825 \dots \times 10^{29}$$

$$\stackrel{?}{=} 1.1726 \dots$$

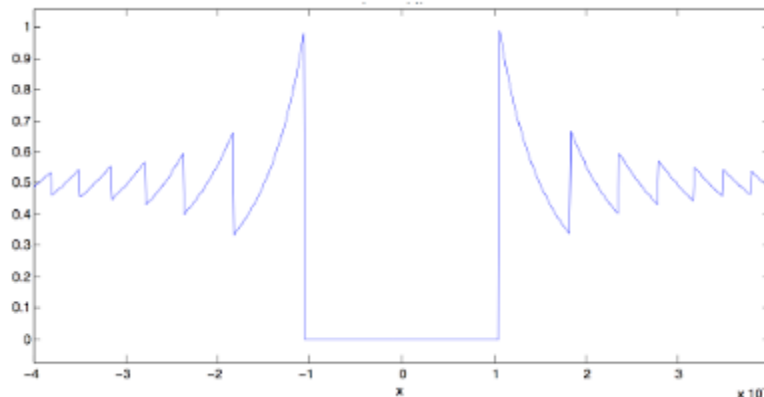
$$\stackrel{?}{=} -0.827396 \dots$$

Catastrophic cancellation

Plot function:

$$f(x) = \frac{1 - \cos x}{x^2}$$

$$-4 \cdot 10^{-8} \leq x \leq 4 \cdot 10^{-8}.$$



`cos (x) = 0.9999999999999999888897769753748434595763683319091796875.`

- **Catastrophic cancellation.** Devastating loss of precision when small numbers are computed from large numbers, which themselves are subject to roundoff error.

Numerical catastrophes

Ariane 5 rocket. [June 4, 1996]

10 year, \$7 billion ESA project exploded after launch.

64-bit float converted to 16 bit signed int.

Unanticipated overflow.



Vancouver stock exchange. [November, 1983]

Index undervalued by 44%.

Recalculated index after each trade by adding change in price.

22 months of accumulated truncation error.



Patriot missile accident. [February 25, 1991]

Failed to track scud; hit Army barracks, killed 28.

Inaccuracy in measuring time in $1/20$ of a second since using 24 bit binary floating point.

Courant-Friedrichs-Lewy condition

- CFL number: numerical stability

$$CFL = \frac{u\Delta t}{\Delta x}$$

$$\underline{CFL < 1}$$

2D:

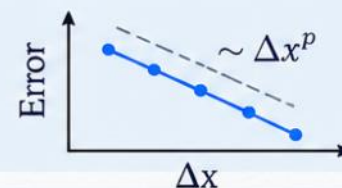
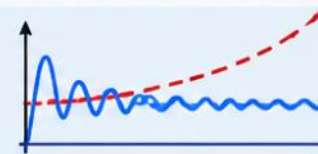
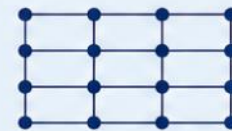
$$\frac{u_x \Delta t}{\Delta x} + \frac{u_y \Delta t}{\Delta y} = C[2D]$$

$$\Delta t = \text{Min}[CFL * u_{ij} \Delta x_{ij}]$$

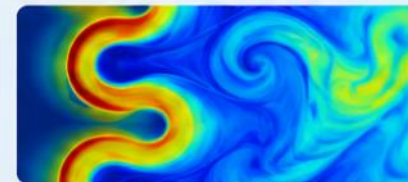
Properties of Numerical Models

A robust simulation has the following properties:

- **Consistency** (discretization correctly represents the underlying equations)
- **Stability** (errors do not grow uncontrollably)
- **Convergence** (solution approaches the true/analytic solution as resolution increases)
- **Conservation** (preserves key physical invariants, e.g. mass, momentum, energy)
- **Boundedness and Accuracy** (solution remains physical and accurate; respects constraints such as positivity, causality, divergence-free fields, etc.)



$$\frac{d}{dt} \int_V Q dV = 0$$



Stability Theory

- Lyapunov stability
- Asymptotic stability
- Exponential stability

1 Lyapunov stability

The equilibrium $x(0) = 0$ is **Lyapunov stable** if, for every $\varepsilon > 0$, there exists $\delta = \delta(\varepsilon) > 0$ such that, if $\|x(0)\| < \delta$, then

$$\|x(t)\| < \varepsilon, \quad \text{for all } t \geq 0.$$

2 Asymptotic stability

The equilibrium $x(0) = 0$ is **asymptotically stable** if it is Lyapunov stable and there exists $\delta > 0$ such that, if $\|x(0)\| < \delta$, then

$$\lim_{t \rightarrow \infty} x(t) = 0.$$

3 Exponential stability

The equilibrium $x(0) = 0$ is **exponentially stable** if it is asymptotically stable and there exist constants $\alpha, \beta > 0$ such that, if $\|x(0)\| < \delta$, then

$$\|x(t)\| \leq \alpha \|x(0)\| e^{-\beta t}, \quad \text{for all } t \geq 0.$$

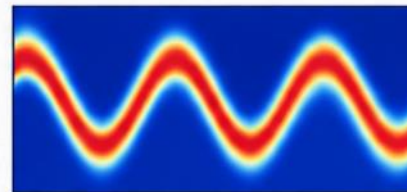
Numerical Stability

A numerical algorithm is **stable** if, for a fixed final time T , the numerical solution remains bounded as the time step $\Delta t \rightarrow 0$.

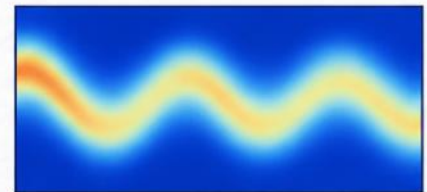
- **Numerical diffusion**

- stabilizes high-frequency modes
- increases with the time step Δt and grid spacing Δx

Small Δt
(stable, low diffusion)



Large Δt
(stable but diffusive)



- **CFL number**

$$CFL = \frac{u \Delta t}{\Delta x}$$

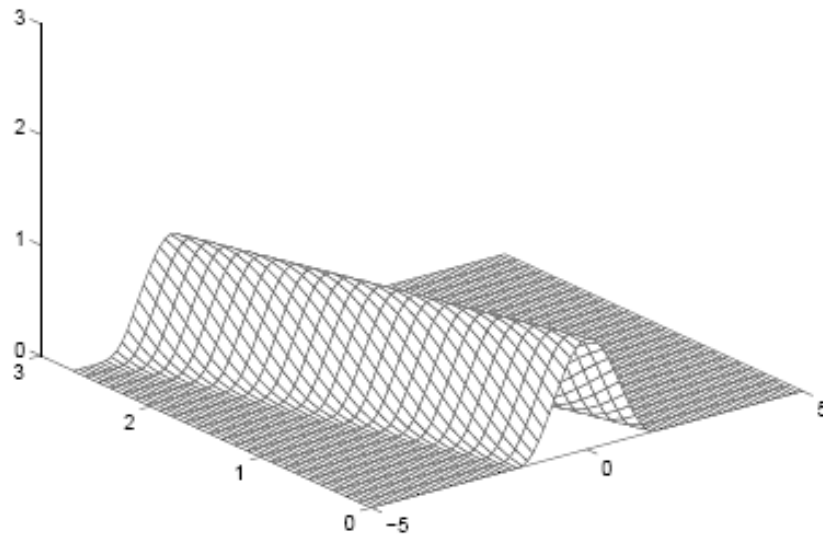
u – characteristic wave speed

Δt – time step

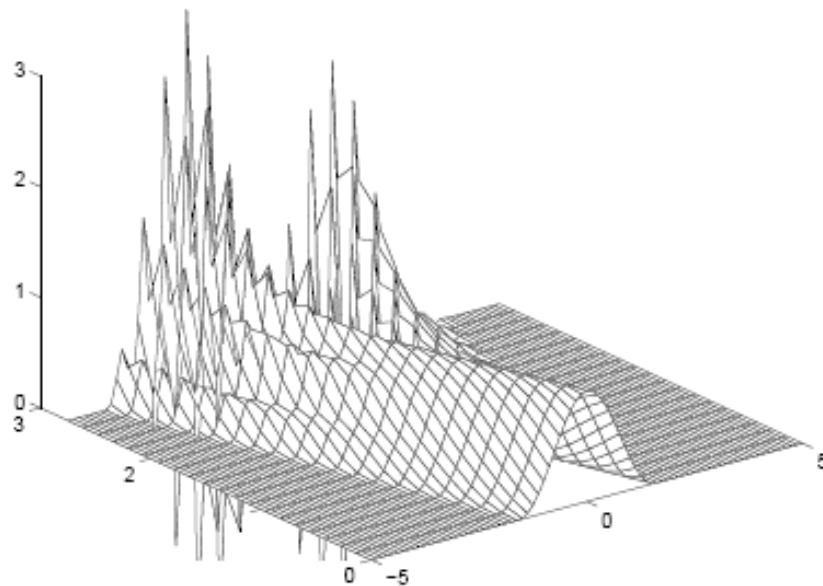
Δx – grid spacing

$CFL < 1$
(stability condition)

Numerical Instability



(a) $\lambda = 0.9$



(b) $\lambda = 1.1$

Courant–Friedrichs–Lewy (CFL) condition

CFL number: numerical stability

1 CFL number

$$CFL = \frac{u \Delta t}{\Delta x}$$

u – characteristic wave speed

Δt – time step

Δx – grid spacing

$$CFL < 1$$

(stability condition)

2 Multi-dimensional case (e.g. 2D)

$$\frac{u_x \Delta t}{\Delta x} + \frac{u_y \Delta t}{\Delta y} \leq C \quad [2D]$$

u_x, u_y – characteristic wave speeds
in x and y directions

$\Delta x, \Delta y$ – grid spacings

C – scheme-dependent constant
(typically $C \sim 1$)

3 Time step determination

$$\Delta t = \min_{i,j} \left[CFL \frac{\Delta x_{ij}}{u_{ij}} \right]$$

u_{ij} – local characteristic speed
(e.g. $|v| + c_s$)

Δx_{ij} – local grid spacing

$CFL \lesssim 1$ – stability requirement

Numerical Methodology

1 Direct Numerical Simulations

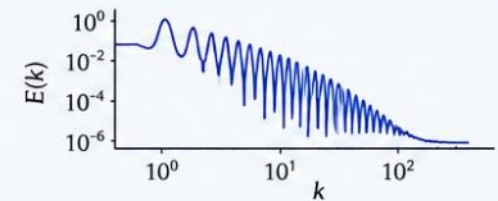
(Godunov, finite difference, finite volume, split, unsplit, etc.)

- Numerical experiment for the full nonlinear system (initial and boundary conditions, resolution studies)



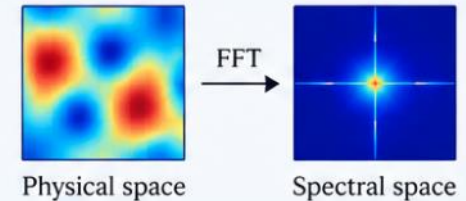
2 Spectral Methods (Fourier, Chebyshev)

- High-accuracy representation for smooth or periodic problems
- Efficient for linear problems and turbulence statistics



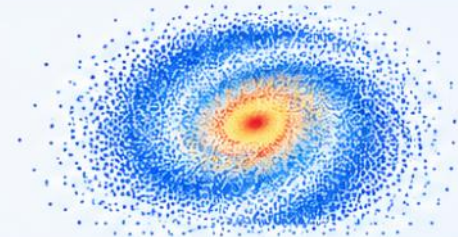
3 Pseudo-Spectral Methods

- Spectral accuracy with nonlinear terms computed in physical space (FFT-based)
- Dealiasing (e.g. 2/3 rule) to remove aliasing errors



4 N-body Methods (SPH)

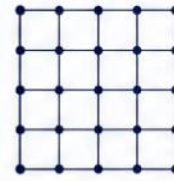
- Lagrangian particle method for collisionless/self-gravitating systems
- Adaptive resolution in high-density regions
- Applications: plasmas, stellar dynamics, large-scale structure



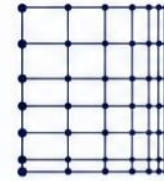
Mesh

Static grids

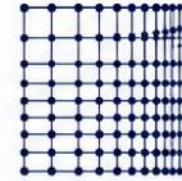
- Uniform grid
- Linearly nonuniform grid
- Complex nonuniformity (Chebyshev, etc.)
- Non-Cartesian grids



Uniform



Linearly
nonuniform



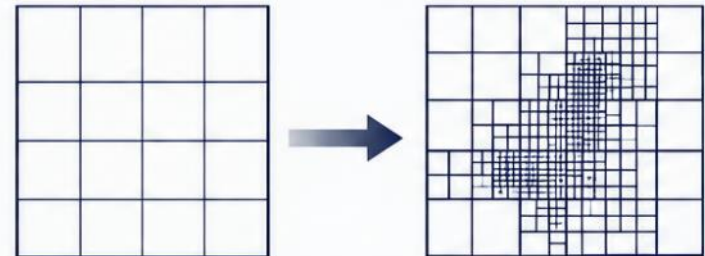
Chebyshev
(nonuniform)



Non-Cartesian
(e.g. polar,
curvilinear)

Dynamical grids

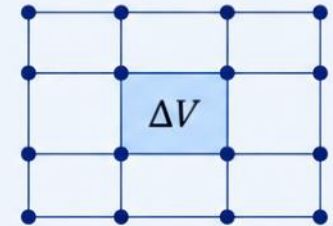
- Adaptive Mesh Refinement (AMR)
 - hierarchical, block-structured grids
 - local refinement based on error estimates or physical criteria
 - efficient resolution of multiscale features (e.g. shocks, turbulence)



Algorithms

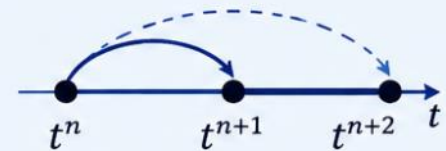
1 Spatial Integration

- Discretize the spatial domain (regular, nonuniform, or adaptive grid)
- Approximate spatial derivatives (finite difference/volume/element)
- Ensure accuracy and conservation (e.g. flux-conservative schemes)



2 Temporal Integration

- Advance the solution in time using an appropriate scheme (e.g. explicit, implicit, or semi-implicit methods)
- Control stability and accuracy (time step, order of the method)

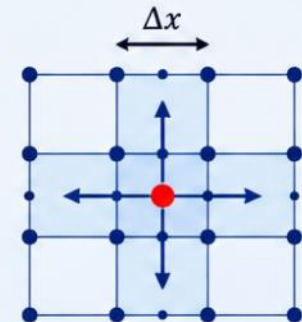


3 Time step determination: CFL condition

- For explicit schemes, the time step is limited by the CFL condition

$$\Delta t \leq C_{CFL} \frac{\Delta x}{|\lambda_{max}|},$$

where Δx is the characteristic grid spacing, $|\lambda_{max}|$ is the maximum wave speed, and $C_{CFL} \leq 1$ is a scheme-dependent constant.



Parallelization

1 Hardware

- Desktop/Laptop (multi-core CPU)
- Beowulf clusters (distributed memory)
- High-performance computing (HPC) systems (multi-node, high-speed interconnect)



PC (multi-core)



Beowulf cluster



HPC system

2 Software

- Parallel programming models and libraries
- Efficient data structures and domain decomposition
- Scalable I/O and checkpointing
- Performance profiling and optimization



MPI

MPI
Message Passing
Interface



OpenMP
Shared-memory
parallelism

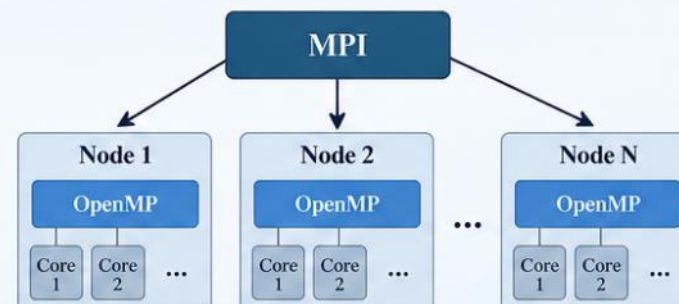


pvm

PVM
Parallel Virtual
Machine

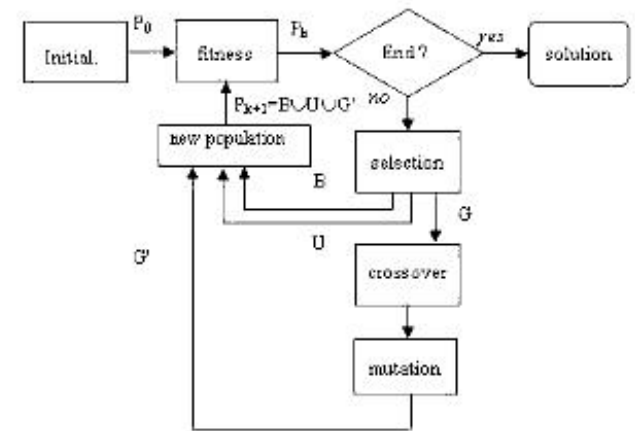
3 Parallelization Approaches

- **MPI** – distributed-memory parallelism (cluster and HPC systems)
- **PVM** – distributed-memory parallelism (legacy systems)
- **OpenMP** – shared-memory parallelism (multi-core CPUs)
- **Hybrid MPI + OpenMP** for large-scale simulations



Pseudocode

Algorithm development



Pseudocode:

Code intended for human reading rather than the machine reading

- *no variable definitions;*
- *no memory management;*
- *no subroutines;*
- *no system-specific code;*

Pseudocode language choice: Matlab

- Avoid Matlab specific functions and simulink

Pseudocode

Pseudocode is intended to be rewritten in low level programming language later

Pseudocode

```
{  
  //IF robot has no obstacle in  
  front THEN  
    // Call Move robot  
    // Add the move command to  
      the command history  
    // RETURN true  
  //ELSE  
    // RETURN false without  
    moving the robot  
  //END IF  
}
```

Java implementation

```
{  
  if (aRobot.isFrontClear())  
  {  
    aRobot.move();  
    cmdHistory.add(RobotAction.MOVE);  
    return true;  
  }  
  else  
  {  
    return false;  
  }  
}
```

PDE classification

Linear second order Partial Differential Equation

$$a \frac{\partial^2}{\partial x^2} \Psi + b \frac{\partial^2}{\partial x \partial y} \Psi + c \frac{\partial^2}{\partial y^2} \Psi + d \frac{\partial}{\partial x} \Psi + e \frac{\partial}{\partial y} \Psi + f \Psi = g$$

Elliptic: $b^2 - 4 a c < 0$

Parabolic: $b^2 - 4 a c = 0$

Hyperbolic: $b^2 - 4 a c > 0$

PDE classification

Elliptic equation:

Poisson equation

Parabolic equation:

Diffusion equation

Hyperbolic equation:

Wave equation

EXAMPLES

Numerical methods: individual treatment

EXAMPLES / EXERCISE

Identify the type of each equation.

1. $u_{xx} + u_{yy} = f(x, y)$

2. $u_t - \kappa u_{xx} = 0, \quad \kappa > 0$

3. $u_{tt} - c^2 u_{xx} = 0$

4. $u_{xx} + 4u_{xy} + u_{yy} = 0$

5. $u_{xx} + 2u_{xy} + u_{yy} = 0$

6. $4u_{xx} + 2u_{xy} + 3u_{yy} = 0$

Blue = elliptic, Green = parabolic, Red = hyperbolic

Conservation Laws

Modelling conservation laws: rewrite a set of equations in the form of the general set of conservation laws (analytically).

1 Conserved quantities: volume integrals

$$Q(t) = \int_V q(\mathbf{x}, t) dV$$

If q satisfies a local conservation law, then the total amount in a fixed volume V is conserved (up to boundary fluxes).

2 Differential form of the continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0$$

$\rho(\mathbf{x}, t)$ – conserved density (e.g. mass, charge)

$\mathbf{V}(\mathbf{x}, t)$ – flux (e.g. velocity field)

3 Mass conservation in total volume

$$M = \int_V \rho dV = \text{const.}$$

For a closed system (no flux through the boundary), the total mass in the volume V is constant in time.

Conservation laws

Generalized form of conservation laws:

$$\frac{\partial \Phi}{\partial t} + \nabla \cdot J = 0$$

Φ – numerical variable

J – numerical flux of the variable Φ

ρ, P, V (physical variables):

primitive variables

Task: *reducing existing system of hyperbolic PDE to the conserving form*

EXAMPLES

Incompressible Hydrodynamics

1 Incompressible HD ($\nu = 0$)

$$\nabla \cdot \mathbf{v} = 0$$

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla) \mathbf{v} = -\frac{1}{\rho_0} \nabla P$$

$\mathbf{v}$ – velocity field
 P – pressure
 ρ_0 – constant density
 (incompressible limit)



2 Conservative form

$$\frac{\partial \mathbf{v}}{\partial t} + \nabla \cdot \mathbf{J} = 0$$

$$\mathbf{J} = \mathbf{v}\mathbf{v} + \frac{P}{\rho_0} \mathbf{I}$$

$\mathbf{v}$ – velocity field
 P – pressure
 $\mathbf{J}$ – flux tensor of $\mathbf{v}$

$$\mathbf{U} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$$

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{J}_x}{\partial x} + \frac{\partial \mathbf{J}_y}{\partial y} + \frac{\partial \mathbf{J}_z}{\partial z} = 0$$

$$\mathbf{J}_x = \begin{pmatrix} v_x^2 + P/\rho_0 \\ v_x v_y \\ v_x v_z \end{pmatrix}$$

$$\mathbf{J}_y = \begin{pmatrix} v_x v_y \\ v_y^2 + P/\rho_0 \\ v_y v_z \end{pmatrix}$$

$$\mathbf{J}_z = \begin{pmatrix} v_x v_z \\ v_y v_z \\ v_z^2 + P/\rho_0 \end{pmatrix}$$

Exercise: Compressible HD

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

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla) \mathbf{v} = -\frac{1}{\rho} \nabla P$$

$$\frac{\partial P}{\partial t} + \mathbf{v} \cdot \nabla P + \gamma P \nabla \cdot \mathbf{v} = 0$$

Conservative form

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{J}_x}{\partial x} + \frac{\partial \mathbf{J}_y}{\partial y} + \frac{\partial \mathbf{J}_z}{\partial z} = 0$$

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho v_z \\ E \end{pmatrix}, \quad E = \frac{P}{\gamma - 1} + \frac{1}{2} \rho v^2.$$

find explicit form of $\mathbf{J}$



end