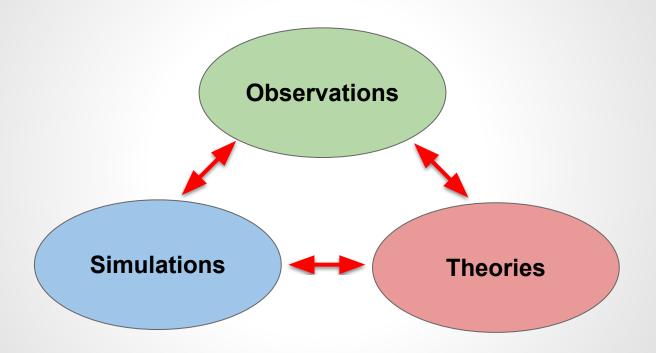
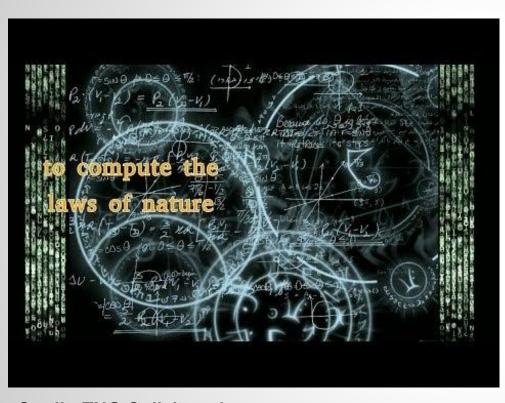
Computational Astrophysics (108-2)

Hsi-Yu Schive (薛熙于) National Taiwan University

Why Simulations?



Example: Large-Scale Structure of the Universe



Credit: TNG Collaboration

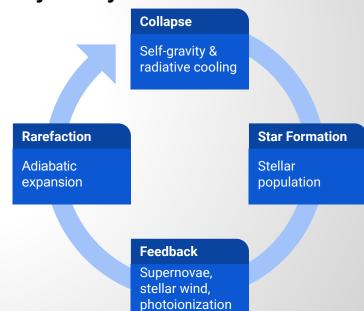
- Illustris cosmological simulation
 - Create mock observations
- Time: 0.3 Myr ~ 13.8 Gyr
- Size: 106.5 Mpc³
- ~10¹⁰ particles & hydro cells
 - DM mass resolution ~ 10⁶ M_o
- Spatial resolution ~ 1 kpc
 - Cover 10⁵ spatial range
- Computing resource: 8192 CPU cores
- Computing time: 1.9x10⁷ CPU hours
 - ~3 months with 8192 CPU cores
- Successor: Illustris TNG
 - https://www.tng-project.org

Example: Milky-Way-Like Galaxy

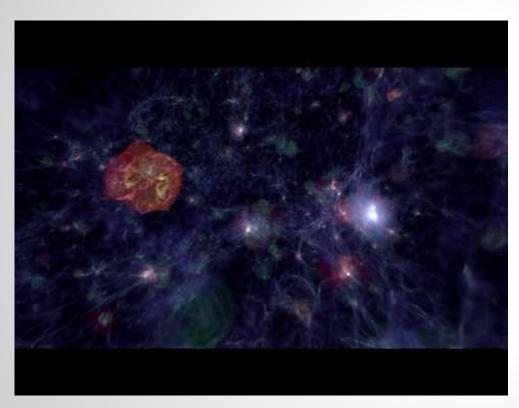


Credit: Advanced Visualization Laboratory at NCSA

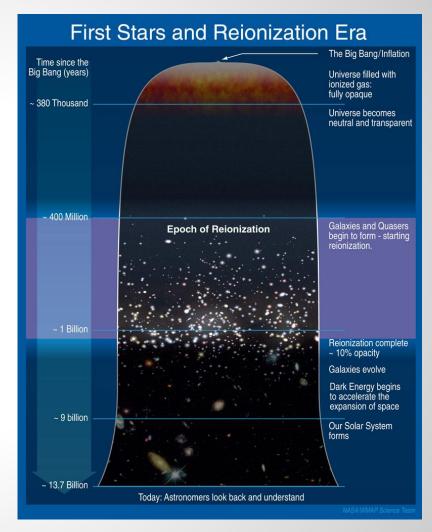
- Isolated disk galaxy simulation
 - Similar to our Milky Way
- Physics cycle



Example: First Stars

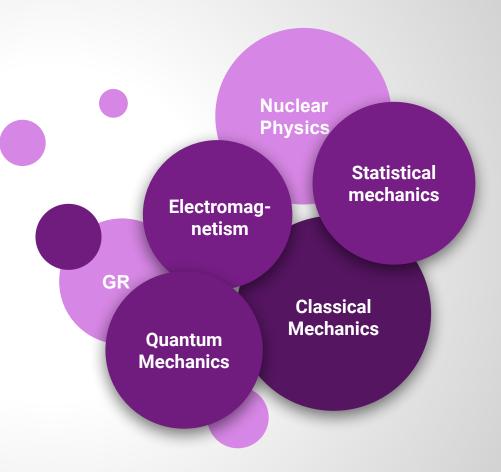


Credit: Renaissance Simulations Laboratory, Advanced Visualization Laboratory at NCSA



Key Physics

- Dark matter
- Hydrodynamics
- Self-gravity
- Magnetic field
- Chemistry
- Radiation transfer
 - Cooling, ionization, etc
- Star formation and evolution
- Feedback
 - Supernovae explosion
 - Stellar wind
 - SMBH/AGN jets
 - 0 ...



Key Techniques

- Numerical algorithms
- Parallel computing
 - CPU/GPU parallelization
- Code co-development
- Data analysis and visualization
- Debugging
- Data sharing

Syllabus

	日期	單元主題
第1週	03/03	Introduction
第2週	03/10	Initial Value Problems
第3週	03/17	Computational Hydrodynamics I
第4週	03/24	Computational Hydrodynamics II
第5週	03/31	Boundary Value Problems
第6週	04/07	Discrete Fourier Analysis
第7週	04/14	N-body: Gravity Evaluation
第8週	04/21	N-body: Orbit Integration
第9週	04/28	HPC: OpenMP & MPI Parallelization I
第10週	05/05	HPC: OpenMP & MPI Parallelization II
第11週	05/12	HPC: OpenMP & MPI Parallelization III
第12週	05/19	HPC: GPU Programming I
第13週	05/26	HPC: GPU Programming II
第14週	06/02	Invited Talk: Core-collapse Supernovae (Prof. Kuo-Chuan Pan from NTHU)
第15週	06/09	Invited Talk: TBD
第16週	06/16	Final Presentation

Course Goals

Numerical algorithms

- Simulations are notorious for "garbage in, garbage out"
- Numerical errors and their origins
- Computational complexity

Parallel Computing

- Astrophysical simulations can be extremely time-consuming
- Single multi-core CPU → multi-CPU → GPU → CPUs + GPUs

Demo

- o Thinking ≠ Learning ⇒ PRACTICE !!
- Runnable demos will be provided for most topics
- In-class practice (bring your laptop!) and homework

Code co-development

- GitHub
- Final project

Grading & TA

- Homework (70%)
 - CEIBA: https://ceiba.ntu.edu.tw
 - Upload within 2 weeks
 - Delay < 1 week: 20% discount
 - Delay ≥ 1 week: zero point
 - NO COPY
- Final project (30%)
 - Team work (3-4 members per group)
 - Upload code to GitHub
 - Demo & oral presentation
- Teaching assistant: Po-Hsun Zheng (zengbs@gmail.com)

Course Prerequisite

- Basic of Python 2 or 3
 - Linux-like system: you should have python installed already
 - Windows: SPYDER may be a good choice
- Basic of C/C++
 - Linux-like system: gnu compiler
 - Windows: try Visual Studio Express
- Contact TA if you need any help or don't have access to a working system

```
temp.py interpolation.py interpolation.py int._py unid_helper.py unid_main.py README.md
Data
spyder
 🖙 .github
                             8 from numpy import cos, linspace, pi, sin, random
  condarecipe
                             9 from scipy.interpolate import splprep, splev
  continuous integration
 img_src
                            11 # XX Generate data for analysis
  rope profiling
                             14 t = linspace(0, 1.75 * 2 * pi, 100)
                            16x = \sin(t)
                            17 v = cos(t)
      init_py
      di_options.py
                            18 z = t
       mac_stylesheet.qss
       mainwindow.pv
                             20 # Add noise
       (ii) restart.pv
                             21 x += random.normal(scale=0.1, size=x.shape)
                             22 y += random.normal(scale=0.1, size=y.shape)
       tour.py
                             23 z += random.normal(scale=0.1, size=z.shape)
                             28 # Spline parameters
                            29 smoothness = 3.0 # Smoothness parameter
                             30 k param = 2 # Spline order
                            31 nests = -1 # Estimate of number of knots needed (-1 = maximal)
    workers
     init_py
                          34 knot_points, u = splprep([x, y, z], s=smoothness, k=k_param, nests=-1)
     dependencies.py
     interpreter.py
                             36 # Evaluate spline, including interpolated points
     atherplugins.pv
                             37 xnew, ynew, znew = splev(linspace(0, 1, 400), knot_points)
     pil patch.pv
     pv3compat.pv
     pyplot.py
                            40 # XX Plot results
  sourier breakpoints
  spyder_io_dcm

✓ 42 # TODO: Rewrite to avoid code smell
  sovder in hdf5
                            43 pylab. subplot(2, 2, 1)
  spyder_profiler
                            44 data, = pylab.plot(x, y, 'bo-', label='Data with X-Y Cross Section')
  spyder_pylint
                            45 fit, = pylab.plot(xnew, ynew, 'r-', label='Fit with X-Y (ross Section')
  .checkignore
                            46 pylab.legend()
  Ciocheck
  ciocopyright.
                            47 pylab.xlabel('x')
   codecov.yml
                            48 pylab.vlabel('v')
```

Quick Taste: Keplerian Motion

```
-1.0 -0.5 0.0
                                                                                0.5
                                                            -1.5
# constants
                                                              t/T = 0.125, error = -2.218e-03
   = 1.0
               # gravitational constant
                                                           1.0
                                                                                           1.0
  = 2.0 # central point mass
dt = 1.0e-2  # time interval for data update
                                                           0.5
                                                                                           0.5
# initial condition
                                                                                           0.0
t
      = 0.0
      = 1.0
                                                          -0.5
                                                                                           -0.5
      = 0.0
                                                          -1.0 -
                                                                                           -1.0
      = (x^{**2} + v^{**2})^{**0.5}
      = 0.0
VX
      = (G*M/r)**0.5
٧V
v abs = (vx**2 + vy**2)**0.5
E0
      = 0.5*v_abs**2 - G*M/r ← Initial total energy for estimating numerical errors later
# plotting parameters
period
                = 2.0*np.pi*r/v_abs
end time
                 = 1.0*period ← Simulate for a single orbit period
nstep per image = 1 \( \bigchip \text{Plotting frequency (i.e., # of updates between two images)} \)
```

Quick Taste: Keplerian Motion

```
def update orbit( i ):
   global t, x, y, vx, vy \leftarrow Use global instead of local variables
   for step in range( nstep_per_image ): ← This loop is simply for reducing the plotting frequency
     calculate acceleration
                                             (which could also be time-consuming!)
           = (x*x + y*y)**0.5
     a_abs = G*M/(r*r)
     ax = -a_abs*x/r \Leftarrow ay = -a_abs*y/r
   update orbit (Euler's method)
     x = x + vx*dt
     y = y + vy*dt \leftarrow Euler's integration: f(t+\Delta t) = f(t) + f'(t)\Delta t + O(\Delta t^2)
     vx = vx + ax*dt
     vy = vy + ay*dt
     update time
     t = t + dt
      if ( t >= end_time ): break ← Stop when reaching the target time
# calculate energy error
       = 0.5*(vx**2 + vy**2) - G*M/r \leftarrow Assuming star mass = 1 for simplicity
   err = (E-E0)/E0
                                                                                         lec01-demo01.py
```

Run lec01-demo01.py

Lessons Learned

- Error ∝ Δt (error per step ∝ Δt²)
- Possible origin of errors?
 - Spatial discretization X
 - Calculating gravity X
 - Updating orbit (Euler's method)
- Validate your code very very very carefully
 - Never trust it without thorough validation
 - How? PHYSICS!
 - Conserved quantity
 - Analytical solution
 - Always ask WHY
 - Real challenge is usually not coding but debugging
- Simulation time

 Δt⁻¹
- Data analysis and visualization is NOT free

Simple Improvement on Orbit Update

Original

```
#
     calculate a(t)
           = (x*x + y*y)**0.5
      a abs = G*M/(r*r)
      ax = -a abs*x/r
      ay = -a abs*y/r
     use v(t) and a(t) to update position
      and velocity by dt
     x = x + vx*dt
     y = y + vy*dt
                       ⇐ Be careful about the
     vx = vx + ax*dt
                           order of update
     vy = vy + ay*dt
```

Revised

```
drift: update position by 0.5*dt
x = x + vx*0.5*dt
y = y + vy*0.5*dt
kick: calculate a(t+0.5*dt) and use that
to update velocity by dt
      = (x*x + y*y)**0.5
a abs = G*M/(r*r)
ax = -a abs*x/r
ay = -a_abs*y/r
vx = vx + ax*dt
    = vy + ay*dt
drift: use v(t+dt) to update position
by another 0.5*dt
x = x + vx*0.5*dt
y = y + vy*0.5*dt
```

Run lec01-demo02.py

Lessons Learned

- Error $\propto \Delta t^2$ (error per step $\propto \Delta t^3$)
- Computational complexity with N particles
 - Position/velocity update: N
 - Computing gravity: N (external gravity), N² (self-gravity)
- Computing time only increases slightly!
 - Position update: 1 → 2 per step
 - Velocity update: still 1 per step
 - Computing gravity: still 1 per step
- Performance
 - Efficient algorithm
 - Scalability
 - Hardware acceleration
 - Extensibility and sustainability

References

Python tutorials

- <u>Programming with Python</u> (good start)
- <u>Learn Python Free Interactive Python Tutorial</u> (online practice)
- The Python Tutorial Python 3.8.2 documentation (official tutorial)
- Python for Beginners (on YouTube)

Online Python interpreters

- https://www.python.org/shell
- https://repl.it/languages
- https://www.onlinegdb.com