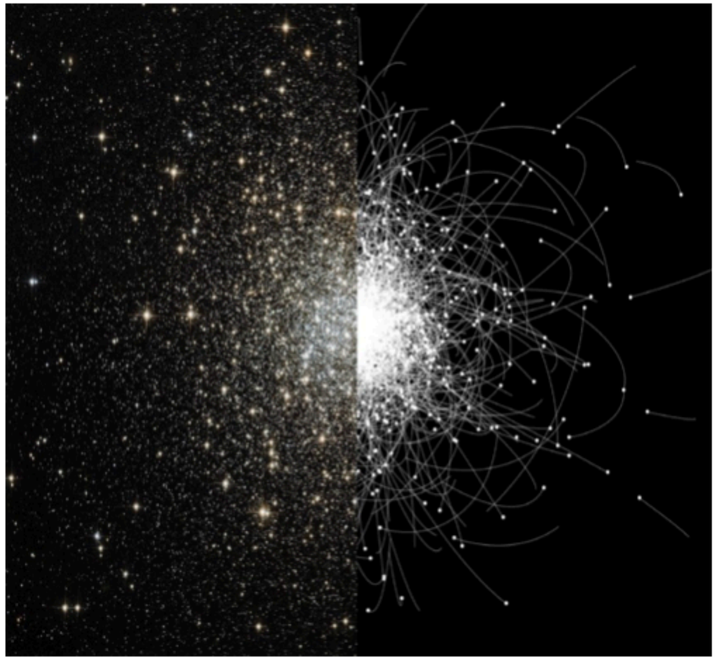
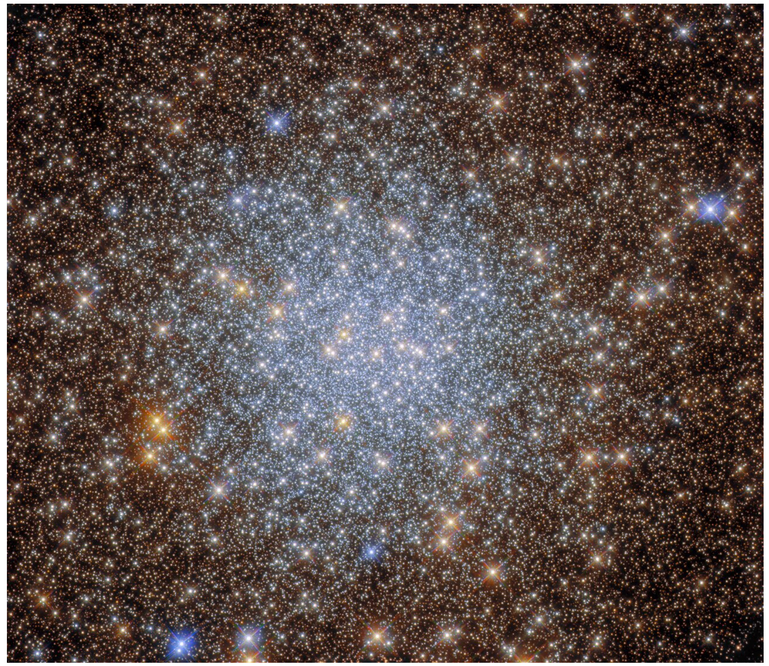
Star Cluster Dynamics and Evolution





Geoplanet Doctoral School Lecture Course (Spring 2024)

Mirek Giersz & Abbas Askar

Nicolaus Copernicus Astronomical Center

Warsaw, Poland

mig@camk.edu.pl askar@camk.edu.pl

Reminder: What do we need to integrate?

The equations of motion to be integrated are

$$\ddot{\mathbf{r}}_i = -G \sum_{j=1,\neq i}^{N} m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{\left|\mathbf{r}_i - \mathbf{r}_j\right|^3}$$

These can be written in the equivalent form

$$\dot{\mathbf{r}}_i = \mathbf{v}_i$$

$$\dot{\mathbf{v}}_i = \mathbf{a}_i = -\sum_{j=1,j\neq i}^{N} G m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{\left|\mathbf{r}_i - \mathbf{r}_j\right|^3}$$

where $\mathbf{r}_i$, $\mathbf{v}_i$ are the position and velocity of the i th particle.

- To solve the ordinary differential equations (2 coupled ODEs)
- Accurate time integration of close encounters is the most difficult part of collisional *N*-body methods
- For collisionless N-body methods force softening alleviates this problem substantially.

Recap: Implementation of direct *N*-body codes for collisional dynamics

- **Direct summation *N*-body approach; "brute force"**
  - NBODYX series of codes: https://people.ast.cam.ac.uk/~sverre/web/pages/nbody.htm (Aarseth 2003)
  - NBODY6++GPU (Wang, Spurzem et al. 2015; 2016): https://github.com/nbody6ppgpu
- **Integration Scheme:**
  - Since close encounters and interactions between stars are important in star clusters → integrator must be high accuracy even on short times scales → 4th order accuracy
  - Expand Taylor series solution for the position and velocities to fourth order in an interval → Hermite integrator

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \Delta t \mathbf{v}_i(t) + \frac{1}{2}(\Delta t)^2 \mathbf{a}_i(t) + \frac{1}{6}(\Delta t)^3 \mathbf{j}_i(t) + \ldots$$

Euler method Algorithm:

$$r(t+\Delta t) = r(t) + v(t)\Delta t$$
$$v(t+\Delta t) = v(t) + a(t)\Delta t$$

Leapfrog algorithm

$$v\left(t+\frac{\Delta t}{2}\right) = v(t) + a(t)\frac{\Delta t}{2}$$

$$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2$$

$$v(t+\Delta t) = v\left(t+\frac{\Delta t}{2}\right) + a(t+\Delta t)\frac{\Delta t}{2}$$

New position is calculated using an extra term proportional to Δt^2
Velocity updated in 2 steps - first half of the time step is taken using the current acceleration and second is taken using the new acceleration

Recap: Hermite Integration: 4th order predictor-corrector

- The algorithm consists of a prediction step:

$$r_p = r(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 + \frac{1}{6}j(t)\Delta t^3$$

$$v_p = v(t) + a(t)\Delta t + \frac{1}{2}j(t)\Delta t^2$$

Taylor series evaluation

- a correction step that makes use of the initial coordinates and the predicted coordinates:

$$r(t+\Delta t) = r(t) + \frac{1}{2}\left(v(t) + v_p\right)\Delta t + \frac{1}{12}\left(a(t) - a_p\right)\Delta t^2$$

$$v(t+\Delta t) = v(t) + \frac{1}{2}\left(a(t) + a_p\right)\Delta t + \frac{1}{12}\left(j(t) - j_p\right)\Delta t^2$$

- j(t) is the jerk which is the time derivative of the acceleration
- a_p is the acceleration calculated using the predicted positions

Calculating the Jerk:

$$\mathbf{a}_i = -G\sum_{j=1,\neq i}^{N} m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{\left|\mathbf{r}_i - \mathbf{r}_j\right|^3}$$

$$\mathbf{j}_i \equiv \dot{\mathbf{a}}_i = -G\sum_{j=1,\neq i}^{N} m_j \left(\frac{\mathbf{v}_i - \mathbf{v}_j}{\left|\mathbf{r}_i - \mathbf{r}_j\right|^3} - 3\frac{\left(\mathbf{v}_i - \mathbf{v}_j\right)\cdot\left(\mathbf{r}_i - \mathbf{r}_j\right)}{\left|\mathbf{r}_i - \mathbf{r}_j\right|^5}\left(\mathbf{r}_i - \mathbf{r}_j\right) \right)$$

Hermite interpolation which approximates the higher accelerating terms by another Taylor series
Trick: Instead of doing more derivatives of jerk, use derivative of predicted values

Recap: Hermite Integration: 4th order predictor-corrector

- The algorithm consists of a prediction step:

$$r_p = r(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 + \frac{1}{6}j(t)\Delta t^3$$

$$v_p = v(t) + a(t)\Delta t + \frac{1}{2}j(t)\Delta t^2$$

Taylor series evaluation

- a correction ste
coordinates an

$$r(t+\Delta t) = r(t) + \frac{1}{2}\left(v(\right.$$

$$v(t+\Delta t) = v(t) + \frac{1}{2}\left(a(\right.$$

4th order Hermite predictor-corrector scheme is 3 step:

1. **predictor step: predicts positions and velocities at 3rd order**
2. **calculation step: calculates acceleration and jerk for the predicted positions and velocities**
3. **corrector step: corrects positions and velocities using the acceleration and jerk calculated in 2**

Calculating the Jerk:

$$\mathbf{a}_i = -G\sum_{j=1,\neq i}^{N} m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{\left|\mathbf{r}_i - \mathbf{r}_j\right|^3}$$

$$\left.\left(\mathbf{v}_i - \mathbf{v}_j\right)\cdot\left(\mathbf{r}_i - \mathbf{r}_j\right)}{\left|\mathbf{r}_i - \mathbf{r}_j\right|^5}\left(\mathbf{r}_i - \mathbf{r}_j\right)\right)$$

erpolation which
e higher accelerating
ther Taylor series
doing more derivatives
of jerk, use derivative of predicted values

- j(t) is the jerk which is the time derivative of the acceleration
- a_p is the acceleration calculated using the predicted positions

Recap: Basic structure of an *N*-body code

1. **Initialisation of $\mathbf{r}_i, \mathbf{v}_i$, tnext $_i$ (update time $t_i + \Delta t_i$), $\mathbf{a}_i, \dot{\mathbf{a}}_i$ for all I**

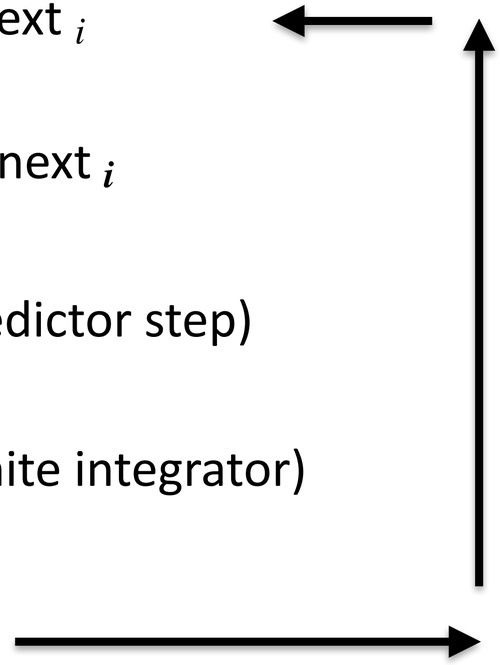
2. Choose i minimising tnext $_i$

3. Extrapolate all $\boldsymbol{r}_j, \mathbf{v}_j$ to tnext $_{\boldsymbol{i}}$

4. Compute new $\mathbf{a}_i, \dot{\mathbf{a}}_i$ (Predictor step)

5. Correct new $\mathbf{r}_i, \mathbf{v}_i$ (Hermite integrator)

6. Compute new tnext $_i$



Note: This is the basic structure of NBODY6, except for the absence of block time steps

Recap: Basic structure of an *N*-body code and time step issues

1. Initialisation of $\mathbf{r}_i, \mathbf{v}_i$, tnext $_i$ (update time $t_i + \Delta t_i$), $\mathbf{a}_i, \dot{\mathbf{a}}_i$ for all I

2. Choose i minimising tnext $_i$

3. Extrapolate all $\boldsymbol{r}_j, \mathbf{v}_j$ to tnext $_i$

4. Compute new $\mathbf{a}_i, \dot{\mathbf{a}}_i$ (Predictor step)

5. Correct new $\mathbf{r}_i, \mathbf{v}_i$ (Hermite integrator)

6. Compute new tnext $_i$

$$\Delta t_i = \eta \frac{a_i}{j_i}$$

- Time step issues:
  - Same time step for all particles?
    - Expensive because a few particles undergo close encounters → force changes more rapidly for them
  - Ideally:
    - Longer time steps for ‘unperturbed' particles
    - Shorter for particles that undergo close encounters
    - Different Δt_i for each particle is expensive and systems lose coherence
  - Block time step scheme: group particles by replacing their individual time steps such that $t/\Delta t_{i,b}$ is an integer (good for synchronization):
    Group together particles which have very similar update times. The extrapolation is shared among them.

Check out Aarseth, Tout & Mardling (eds): *The Cambridge N-Body Lectures (*2008) for details

Recap: Basic structure of an *N*-body code and time step issues

1. Initialisation of $\mathbf{r}_i, \mathbf{v}_i$, tnext $_i$ (update time $t_i + \Delta t_i$), $\mathbf{a}_i, \dot{\mathbf{a}}_i$ for all I

2. Choose i minimising tnext $_i$

3

4

5

6

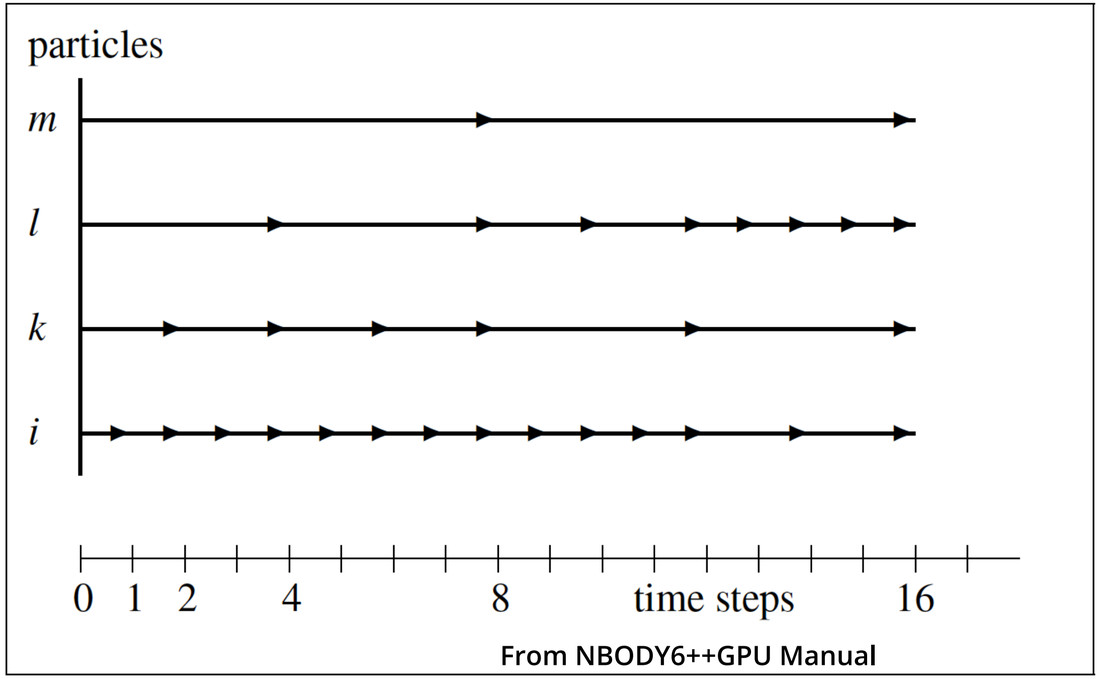


Figure 9.1: Block time steps exemplary for four particles.

$$\Delta t_i = \eta \frac{a_i}{j_i}$$

- Time step issues:
  - Same time step for all particles?
    - Expensive because a few particles undergo close encounters → force changes more rapidly for them
  - Ideally:
    - Longer time steps for ‘unperturbed' particles
    - Shorter for particles that undergo close encounters
    - Different Δt_i for each particle is expensive and systems lose coherence
  - Block time step scheme: group particles by replacing their individual time steps such that $t/\Delta t_{i,b}$ is an integer (good for synchronization):
    Group together particles which have very similar update times. The extrapolation is shared among them.

Check out Aarseth, Tout & Mardling (eds): *The Cambridge N-Body Lectures (*2008) for details + NBODY6++GPU Manual

Recap: Basic structure of an *N*-body code and time step issues

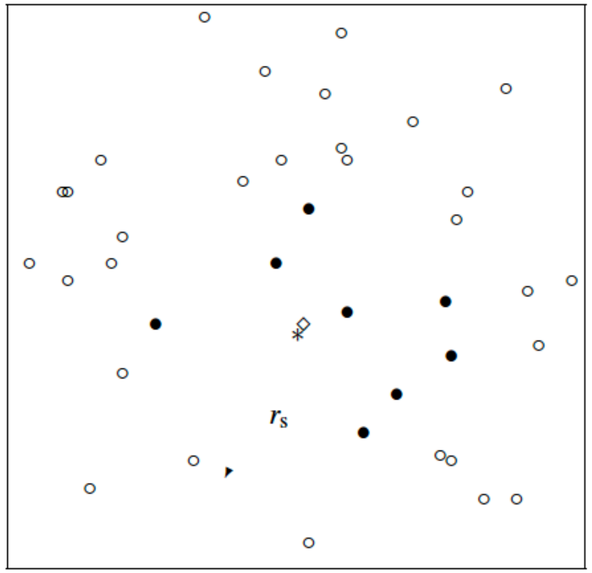


Figure 10.1: Illustration of the neighbour scheme for particle i marked as the asterisk (after [2]).

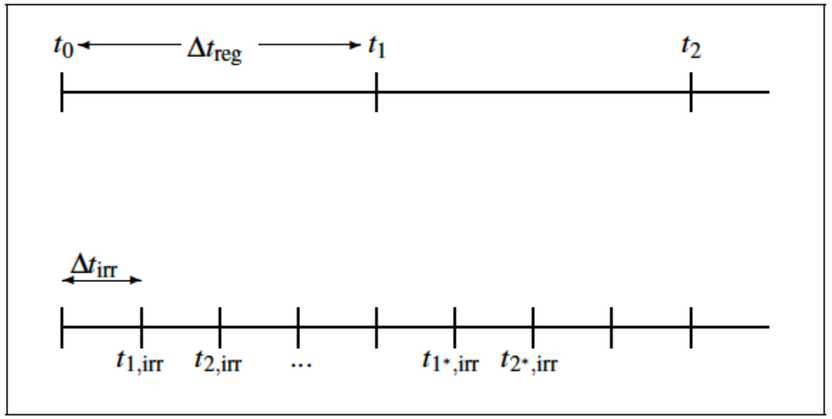


Figure 10.2: Regular and irregular time steps (after [22]).

From NBODY6++GPU Manual

- Neighbour Scheme (Ahmad & Cohen 1973)
  - Time step determined by nearest neighbour
  - Few near neighbours→ force due to them can be computed frequently with little effort (the "irregular force")
  - force due to the more numerous non-neighbours (the "regular force") fluctuates more slowly, and can be computed with a longer time step
  - Requires keeping a list of neighbours

Check out Aarseth, Tout & Mardling (eds): *The Cambridge N-Body Lectures (*2008) for details + NBODY6++GPU Manual

Recap: Basic structure of an *N*-body code and time step issues

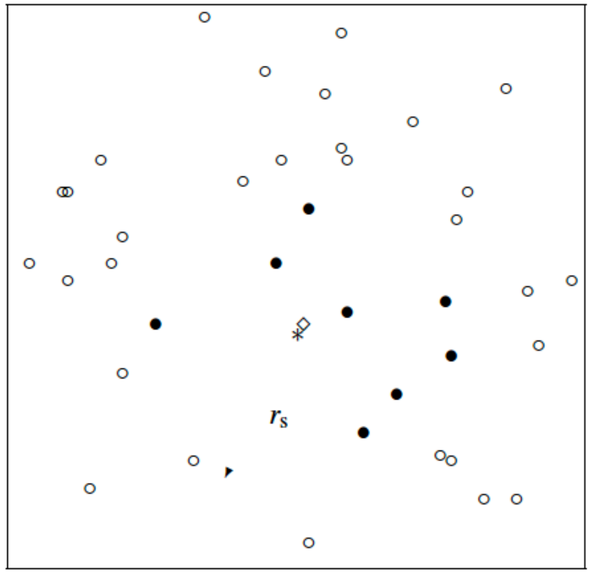


Figure 10.1: Illustration of the neighbour scheme for particle i marked as the asterisk (after [2]).

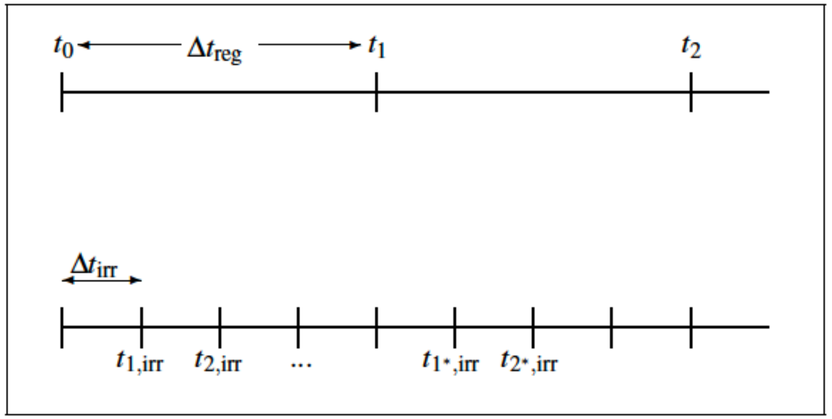


Figure 10.2: Regular and irregular time steps (after [22]).

From NBODY6++GPU Manual

- Neighbour Scheme (Ahmad & Cohen 1973)
  - Time step determined by nearest neighbour
  - Few near neighbours→ force due to them can be computed frequently with little effort (the "irregular force")
  - force due to the more numerous non-neighbours (the "regular force") fluctuates more slowly, and can be computed with a

The **irregular forces** are those fast-changing forces from nearby particles inside the neighbour sphere $\mathbf{a}_i = \mathbf{a}_{i,\,\rm irr} + \mathbf{a}_{i,\,\rm reg}$

The **regular forces** are from the distant particles outside of the neighbour sphere, which change more slowly

Recap: Basic structure of an *N*-body code and time step issues

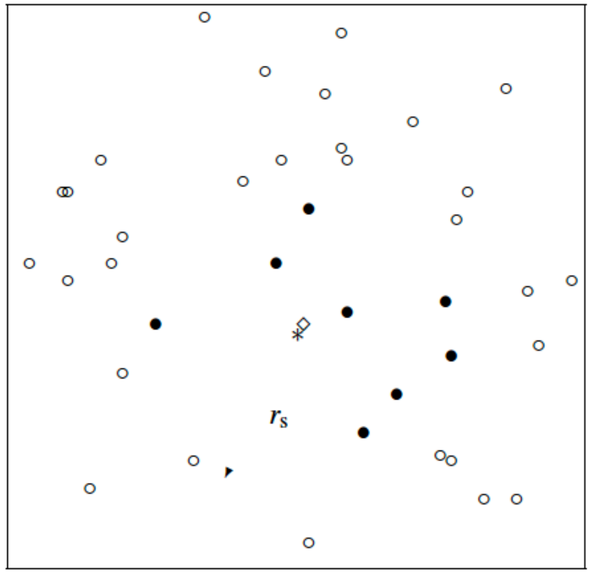


Figure 10.1: Illustration of the neighbour scheme for particle i marked as the asterisk (after [2]).

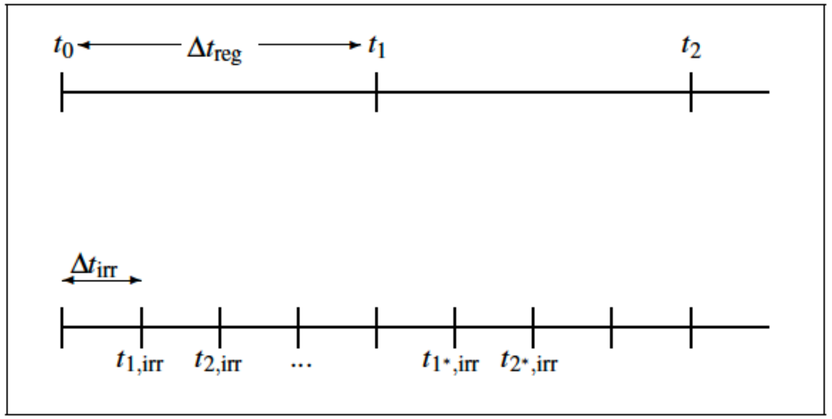


Figure 10.2: Regular and irregular time steps (after [22]).

From NBODY6++GPU Manual

- Neighbour Scheme (Ahmad & Cohen 1973)
  - Time step determined by nearest neighbour
  - Few near neighbours→ force due to them can be computed frequently with little effort (the "irregular force")
  - force due to the more numerous non-neighbours (the "regular force") fluctuates more slowly, and can be computed with a longer time step
  - Requires keeping a list of neighbours

Check out Aarseth, Tout & Mardling (eds): *The Cambridge N-Body Lectures (*2008) for details + NBODY6++GPU Manual

Summary: Neighbour scheme and hierarchical time-step scheme

- Both have in common that they are centered on one particle i
- Both distinguish between nearby and remote stars, and they save computational time
- What is the fundamental difference between them?
  - Neighbour scheme is a spatial hierarchy, which avoids a frequent force calculation of the remote particles → their totality provides a smooth potential which does not vary so much → potential is superposed by fluctuating peaks of close-by stars (irregular force)
  - Time step scheme exhibits the temporal behaviour of the intervals for re-calculation of the full force in order to maintain the exactness of the trajectory; time steps chosen too small slow down the advancing calculation losing the computer's efficiency

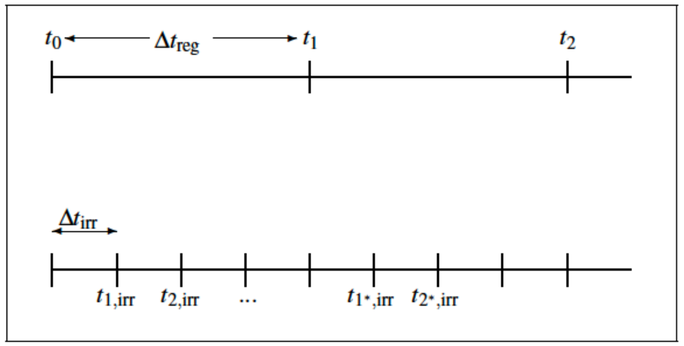


Figure 10.2: Regular and irregular time steps (after [22]).

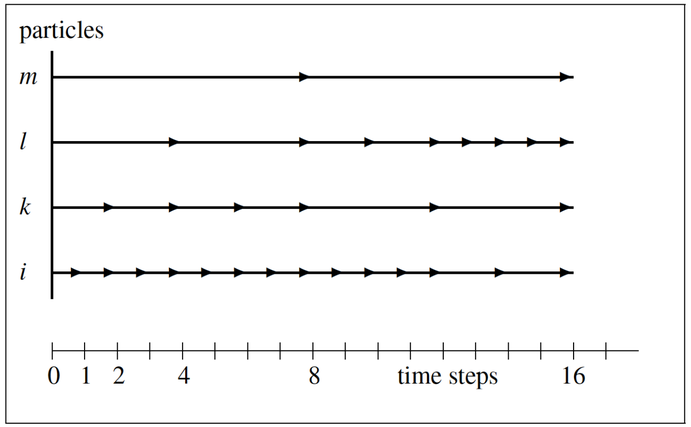


Figure 9.1: Block time steps exemplary for four particles.

Regularization

- Stellar systems are characterized by a huge dynamical range in distance and time scales.
  - The time scale varies e.g. in a star cluster from orbital periods of binaries of some hours/days up to the relaxation of a few hundred million years, or even billions of years.
- A key problem when modelling collisional dynamics is dealing with the divergence in the force for $\boldsymbol{r}_i \rightarrow \boldsymbol{r}_j$

$$\ddot{\mathbf{r}}_i = -G \sum_{j=1,\neq i}^{N} m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^3}$$

Recap: Softening

- The equation of motion is

$$\ddot{\mathbf{r}}_i = -G \sum_{j=1,\neq i}^{N} m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{\left|\mathbf{r}_i - \mathbf{r}_j\right|^3}$$

- Singularity as $\left|\mathbf{r}_i - \mathbf{r}_j\right| \to 0$ → can cause very small time steps
- Replace denominator with $\left(\left|\mathbf{r}_i - \mathbf{r}_j\right|^2 + \varepsilon^2\right)^{3/2}$

$$\mathbf{F}_i = \sum_{i \neq j} \frac{G m_i m_j}{\left|\mathbf{r}_i - \mathbf{r}_j\right|^2 + \epsilon^2}$$

- ε is a small constant: softening parameter
- **This approximation may be justifiable if close encounters between particles are unimportant (e.g., galaxy dynamics)**
- **Not necessarily good for modelling star clusters → physically eliminates formation of binaries with** $r < \epsilon$
- **Hard binaries (very small separation) are an important source of energy in clusters**

Regularization: binaries/close encounters in *N*-body codes for collisional dynamics

- Stellar systems are characterized by a huge dynamical range in distance and time scales.
  - The time scale varies e.g. in a star cluster from orbital periods of binaries of some hours/days up to the relaxation of a few hundred million years, or even billions of years.
- A key problem when modelling collisional dynamics is dealing with the divergence in the force for $\boldsymbol{r}_i \rightarrow \boldsymbol{r}_j$

$$\ddot{\mathbf{r}}_i = -G \sum_{j=1,\neq i}^{N} m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^3}$$

  - Hermite 4th order behaviour: Time step reduction
  - smaller $\Delta t \rightarrow$ more steps to reach the same time $\rightarrow$larger numerical errors
- To avoid this problem, collisional *N*-body codes introduce regularisation for particles that move on tightly bound orbits or for when $\boldsymbol{r}_i \rightarrow \boldsymbol{r}_j$

Regularization: a way of handling binaries/close encounters

- Mathematical trick → remove the singularity in the Newtonian law of gravitation for two particles which approach each other arbitrarily close
  - change of variables
  - without affecting the physics (different from softening)
- Kustaanheimo-Stiefel (KS) regularization: for binaries and 3-body encounters
  - Change from coordinates to offset coordinates: CM and relative particle

$$x_{CM} = \frac{m_1 x_1 + m_2 x_2}{m_1 + m_2} \qquad x_{rel} = x_1 - x_2$$

  - Kepler orbit is transformed into a harmonic oscillator
    - Significantly reduces the number of steps needed to integrate the orbit and reduces round-off error

One dimensional regularization

- Consider unperturbed binary motion:

$$\ddot{r} = -\frac{1}{r^2}$$

- where we have scaled to units such that $G\left(m_i + m_j\right) = 1$. Introduce new variables z, τ (transformed coordinate and time) such that

$$r = z^2$$
$$\frac{dt}{d\tau} = r$$

Since $r = z^2, \dot{r} = 2z\dot{z}$

Since $dt/d\tau = r$,

$\dot{r} = 2zz'\frac{d\tau}{dt}$ where , means $d/d\tau$

$= \frac{2zz'}{r}$ since $dt/d\tau = r$

$= \frac{2z'}{z}.$

Derivation based on ‘Computation and astrophysics of the *N*-body problem’ by Douglas Heggie

One dimensional regularization

$$\dot{r} = \frac{2z'}{z} \qquad \ddot{r} = -\frac{1}{r^2} \qquad r = z^2 \qquad \frac{dt}{d\tau} = r$$

- Differentiating again with respect to time gives similarly

$$\ddot{r} = \frac{2z''}{z^3} - 2\frac{z'^2}{z^4}$$
$$= -\frac{1}{r^2} \text{ (equation of motion)}$$
$$= -\frac{1}{z^4}$$

We have $\frac{2z''}{z^3} - 2\frac{z'^2}{z^4} = -\frac{1}{z^4}$

Hence
$$z'' = \frac{z'^2}{z} - \frac{1}{2z}$$
$$= \frac{1}{2}z\left(\frac{2z'^2}{z^2} - \frac{1}{z^2}\right)$$
$$= \frac{1}{2}z\left(\frac{1}{2}\dot{r}^2 - \frac{1}{r}\right)$$
$$= \frac{1}{2}hz$$

$$h = \frac{1}{2}\dot{r}^2 - \frac{1}{r}$$

One dimensional regularization

- Now
$$h = \frac{1}{2}\dot{r}^2 - \frac{1}{r}$$
is the energy of the binary (per unit [reduced] mass); the kinetic energy is $\dot{r}^2/2$, the potential energy is $-1/r$. Thus in the transformed equation of motion
$$z'' = \frac{1}{2}hz$$
the coefficient of z is constant. This is the simple harmonic oscillator equation
- Transformed the one-dimensional Kepler problem (which is singular at r = 0) into the simple harmonic oscillator equation, which is regular everywhere.

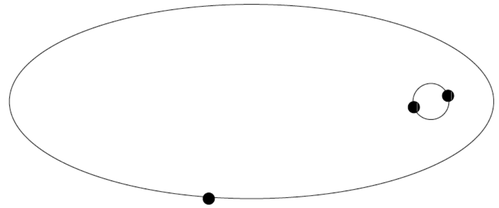
$$\ddot{r} = -\frac{1}{r^2} \qquad\qquad z'' = \frac{1}{2}hz$$

- 3D version of this is called KS regularisation

See Heggie & Hut (2003), chapter 15 for more details

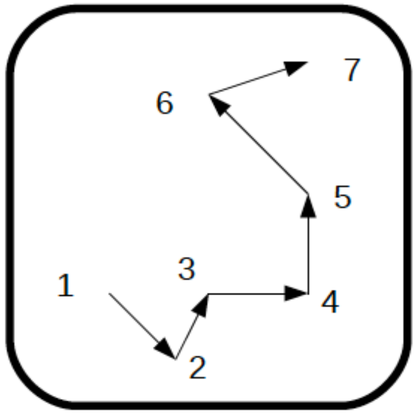
Regularization: a way of handling close encounters

- Hierarchical triples are binaries constantly perturbed by a third body: there is a procedure called “slow-down” which follows secular perturbations with (much) larger time step



Credit: Lecture slides from Douglas Heggie

- CHAIN regularization (Mikkola & Aarseth 1993: for small N-body systems (non-hierarchical triples, quadruples)
  - Calculate distances between an active object (e.g. binary) and the closest neighbours
  - find vectors that minimize the distances → use these vectors (“chain coordinates”)
  - to change coordinates and make suitable changes of time coordinates → calculate forces with new coordinates



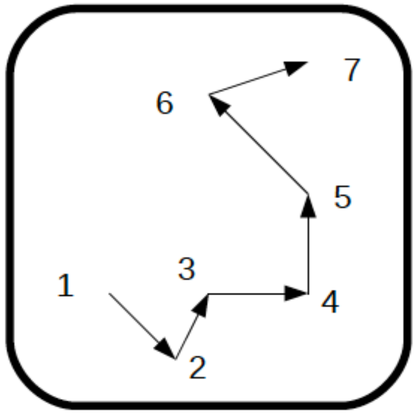
Credit: Mapelli lectures on *N*-body techniques in astrophysics

Regularization: a way of handling close encounters

- Mathematical trick → remove the singularity in the Newtonian law of gravitation for two particles which approach each other arbitrarily close (change of variables without affecting the physics, different from softening)
- Kustaanheimo-Stiefel (KS) regularization: for binaries and 3-body encounters
  - Change from coordinates to offset coordinates: CM and relative particle
  - Kepler orbit is transformed into a harmonic oscillator
    - Significantly reduces the number of steps needed to integrate the orbit and reduces round-off error
- CHAIN regularization (Mikkola & Aarseth 1993: for small N-body systems
  - Calculate distances between an active object (e.g. binary) and the closest neighbours
  - find vectors that minimize the distances → use these vectors (“chain coordinates”)
  - to change coordinates and make suitable changes of time coordinates → calculate forces with new coordinates

$$x_{CM} = \frac{m_1 x_1 + m_2 x_2}{m_1 + m_2}$$

$$x_{rel} = x_1 - x_2$$



Credit: Mapelli lectures on *N*-body techniques in astrophysics

Errors and quality control

- How do we know if the results from an N-body simulation are correct?
  - No exact solutions
- Check conserved quantities
  - Total momentum
  - Total angular momentum
  - Total energy (most sensitive)
- Growth of errors is exponential
  - After a short time the positions and velocities of particles are wrong
  - Assume that the statistical properties of the simulation are correct

Basic structure of a collisional *N*-body code: complexity

1. Initialisation of $\mathbf{r}_i, \mathbf{v}_i$, tnext $_i$ (update time $t_i + \Delta t_i$), $\mathbf{a}_i, \dot{\mathbf{a}}_i$ for all I — Done Once

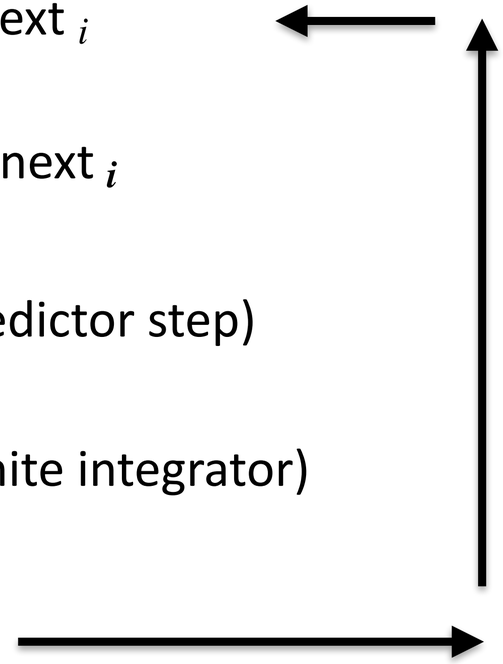
2. Choose i minimising tnext $_i$ — $\propto N$

3. Extrapolate all $\boldsymbol{r}_j, \mathbf{v}_j$ to tnext $_i$ — $\propto N$

4. Compute new $\mathbf{a}_i, \dot{\mathbf{a}}_i$ (Predictor step) — $\propto N$

5. Correct new $\mathbf{r}_i, \mathbf{v}_i$ (Hermite integrator)

6. Compute new tnext $_i$



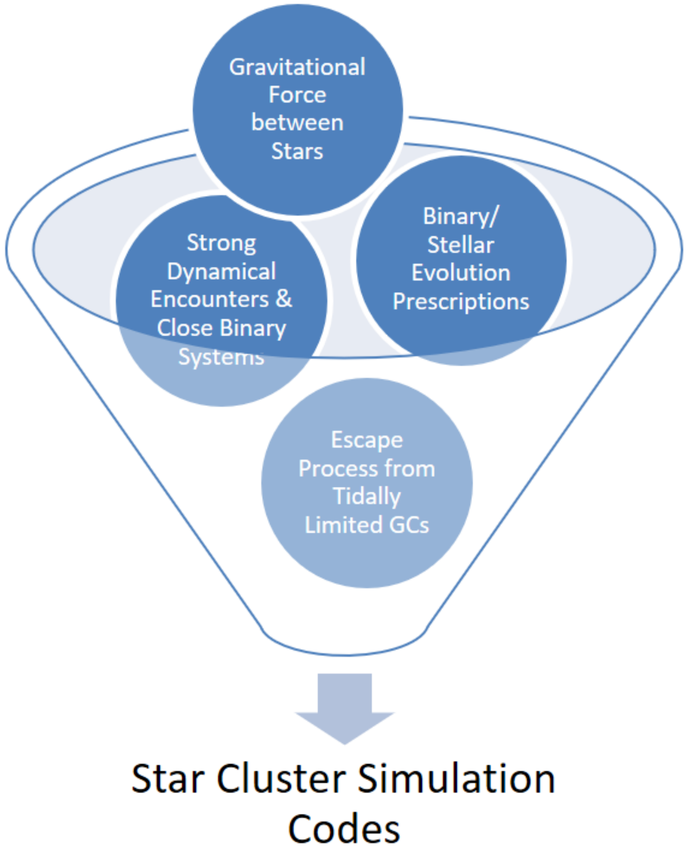
For each time step Δt, the computational effort is approximately proportional to N.

Basic structure of a collisional *N*-body code: complexity

- Typical time step is $\Delta t \sim \frac{r}{v}$, where v is typical speed and r is typical distance to nearest neighbour
- The system has volume of order R^3, where R is the virial radius. Divided among N stars, each star occupies volume of order R^3/N. Hence $r \sim R/N^{1/3}$
- In *N*-body units (Hénon units) $v \sim 1, R = 1$, and so $\Delta t \sim N^{-1/3}$
- Each time step takes of order N operations, and there are N particles. Hence the computational effort per Hénon time unit is of order $N \times N \times N^{1/3} = N^{7/3}$

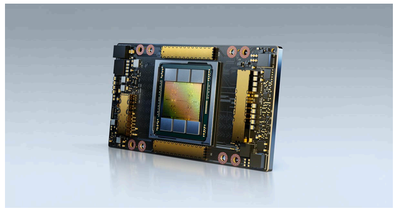
Simulating a globular cluster with NBODY6

- Includes treatment of stellar and binary evolution for various metallicities
  - Based on population synthesis codes → SSE/BSE (Hurley et al. 2002)
    →numerous updates made to these prescriptions (Kamlah et al. 2022)
- Includes effects of external forces (the gravitational acceleration due to the Galaxy)
- Includes rudimentary treatment of collisions between stars (sticky particle approximation)
- Main issue:
  - Computation is slow → main bottleneck → computing new acceleration and jerk
  - Realistic globular clusters can take up to a year or more to evolve on supercomputers + GPU acceleration



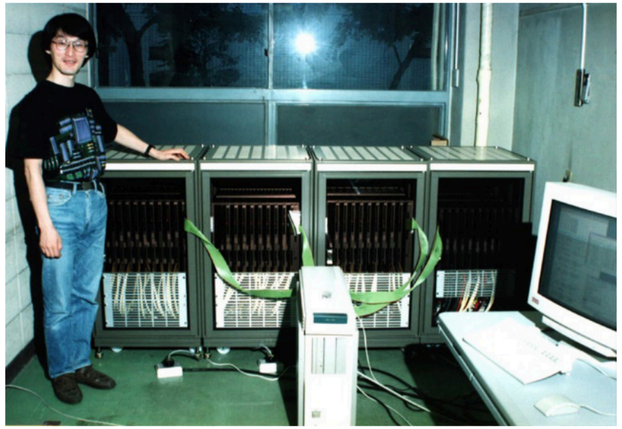
Hardware acceleration of *N*-body *codes*

- Graphics Processing Units (GPUs):
  - Efficient matrix operations (used for 3D renderings)
  - **Architecture: Single instruction on multiple data threads**
    - ***Single operation: Acceleration and jerk calculation on many pairs of particles***
    - ***Each interparticle force between a pair is independent of other pairs***
  - Can be programmed with CUDA
  - Potential for scientific research/AI/Data analytics being realised (mid 2000s to present-day)
- GRAvity PipE (GRAPE): a hardware implementation of Newtonian pair-wise force calculations between particles in a self-gravitating N-body system: https://www.ias.edu/ids/~piet/act/comp/hardware
  - Made by researchers working on star cluster evolution in Tokyo (D. Sugimoto & Jun Mukino et al. from 1989 to 2001)
  - 2001: GRAPE-6 system at 48 TFLOP (for comparison Sony PS5 is 10.28 TFLOPS)



NVIDIA A100 GPUs

NVIDIA Titan V: 110



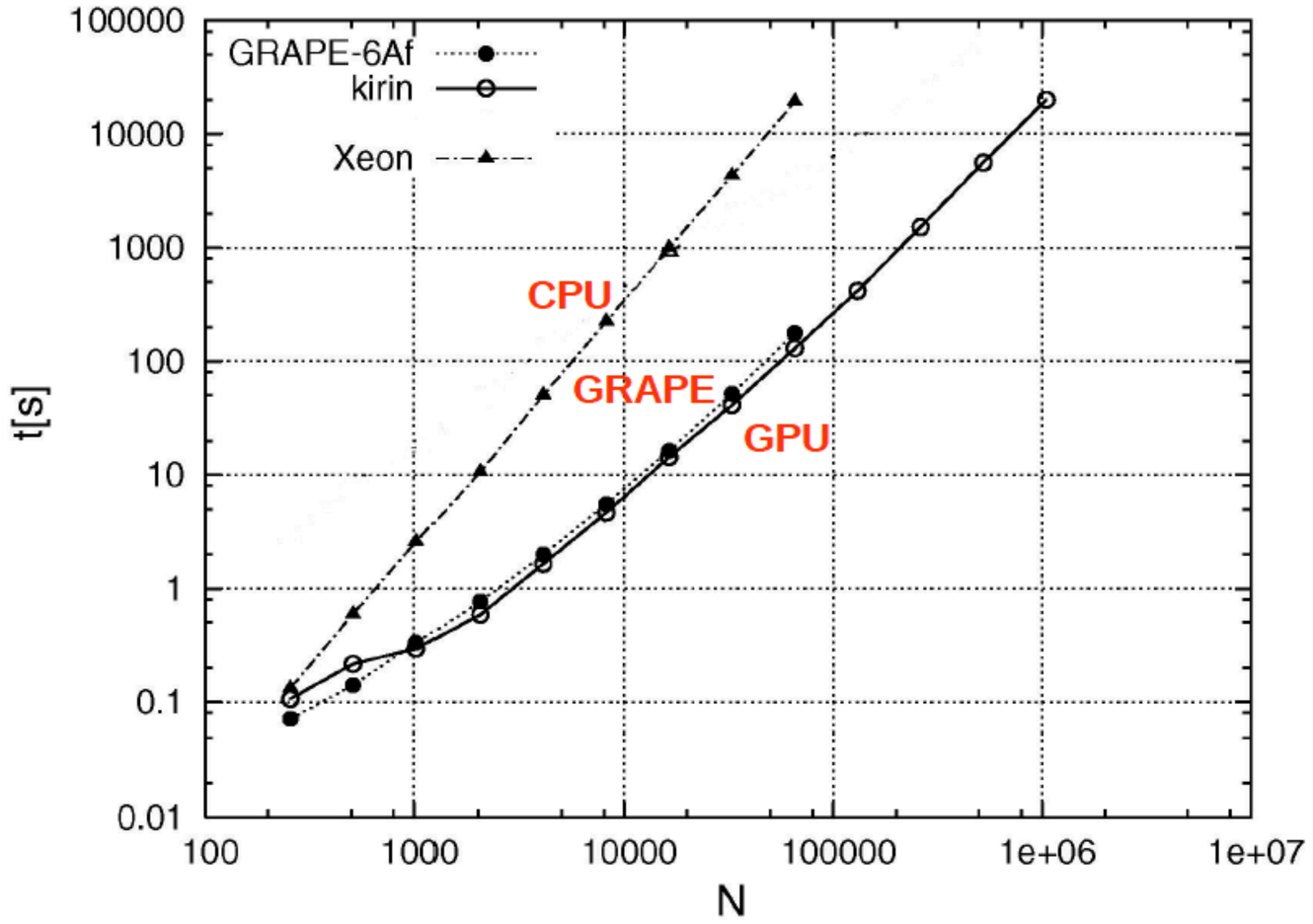
J. Makino with GRAPE-4, ca.1995

Hardware acceleration of *N-body codes*

- Graphics Processing U
  - Efficient matrix operatior
  - **Architecture: Single ins**
    - ***Single operation: Accele***
    - ***Each interparticle force***
  - Can be programmed with
  - Potential for scientific res

- GRAvity PipE (GRAPE): pair-wise force calcula body system: https://w
  - Made by researchers wor Jun Mukino et al. from 19
  - 2001: GRAPE-6 system at

In 2004-2008, researchers found that GPUs are at least as fast as GRAPES for direct N-body codes (Portegies Zwart et al. 2007; **Belleman et al. 2008**; Gaburov et al. 2009)



NVIDIA Titan V: 110

J. Makino with GRAPE-4, ca.1995

NBODYX Versions

A brief comparison of the code versions:

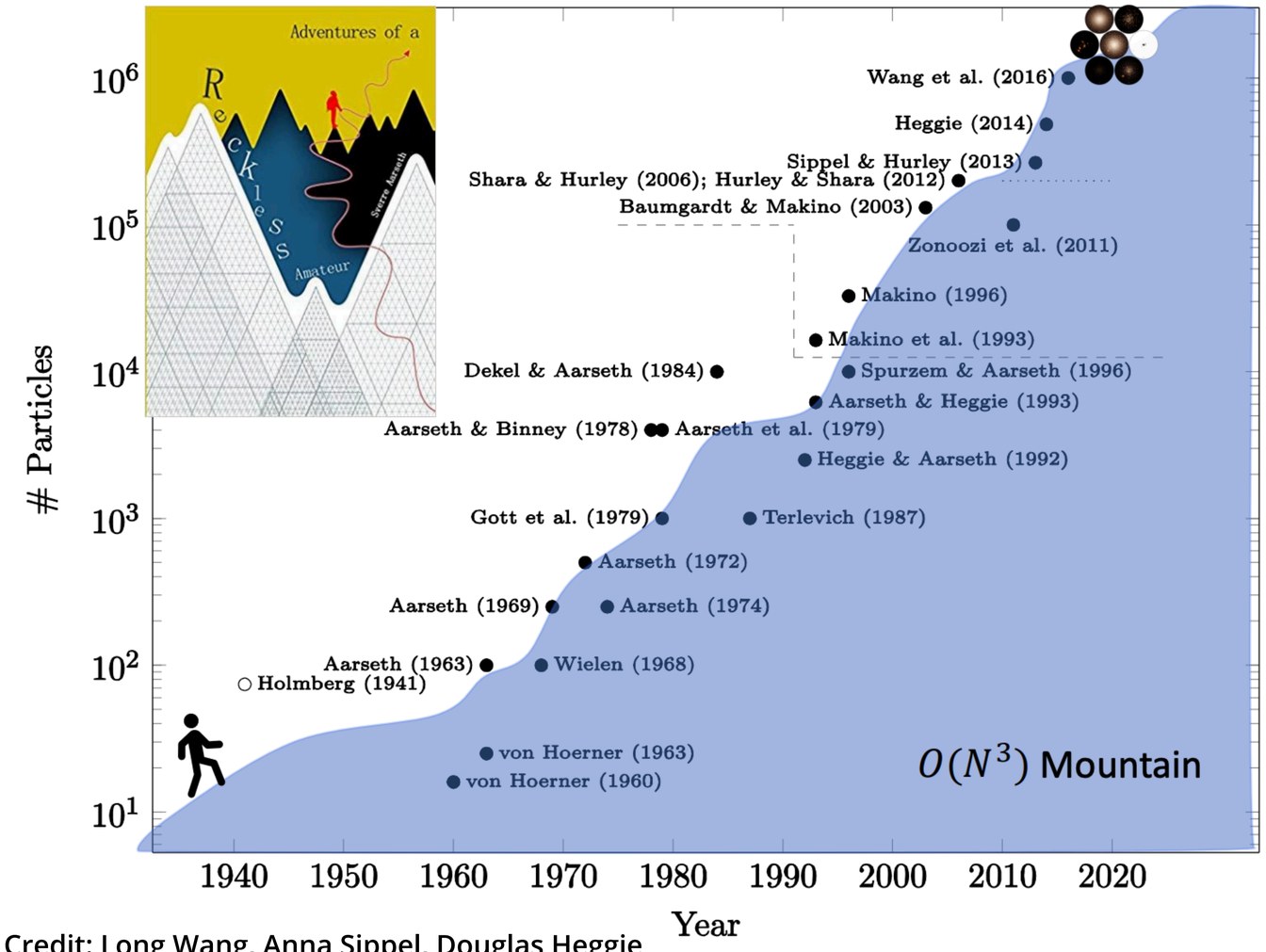
ITS: Individual time–steps
ACS: Neighbour scheme (Ahmad–Cohen scheme) with block time–steps
KS: KS–regularization of few-body subsystems
HITS: Hermite scheme integration method combined with hierarchical block time steps
PN: Post-Newtonian terms
CC: Classical regularized chain
AR: Algorithmic regularization chain
MPI: Message Passing Interface, multi-node multi-CPU parallelization
GPU: use of GPU acceleration (if also MPI: multi-node many GPU)
r: restricted Post-Newtonian, only orbit-averaged energy loss by grav. radiation

	ITS	ACS	KS	HITS	PN	CC	AR	MPI	GPU
NBODY1	✓								
NBODY2		✓		✓					
NBODY3	✓		✓						
NBODY4			✓	✓					
NBODY5	✓	✓	✓						
NBODY6		✓	✓	✓	r	✓	✓		
NBODY6GPU		✓	✓	✓	✓	✓	✓		✓
NBODY6++		✓	✓	✓		✓		✓	
NBODY6++GPU		✓	✓	✓	r	✓		✓	✓
NBODY7		✓	✓	✓	r		✓		✓

NBODY6++GPU Manual
https://www.overleaf.com/read/hcmxcyffjkzq

Developments in Direct *N*-body simulations

Direct N-body is a mountain challenge



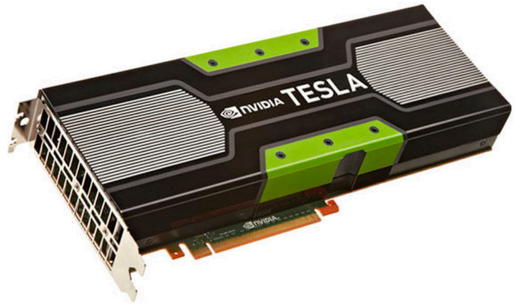
Credit: Long Wang, Anna Sippel, Douglas Heggie

The DRAGON simulations: globular cluster evolution with a million stars

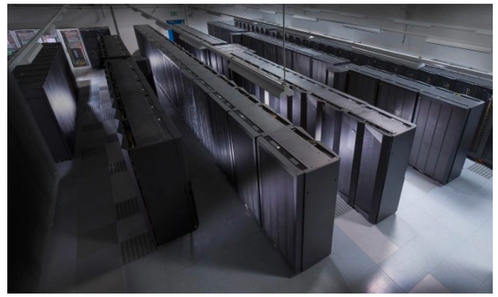
Long Wang,[1,2★] Rainer Spurzem,[3,4,1★] Sverre Aarseth,[5] Mirek Giersz,[6] Abbas Askar,[6] Peter Berczik,[3,4,7] Thorsten Naab,[8] Riko Schadow[8] and M. B. N. Kouwenhoven[1,2]

[1]*Kavli Institute for Astronomy and Astrophysics, Peking University, Yiheyuan Lu 5, Haidian Qu, 100871 Beijing, China*
[2]*Department of Astronomy, School of Physics, Peking University, Yiheyuan Lu 5, Haidian Qu, 100871 Beijing, China*
[3]*National Astronomical Observatories and Key Laboratory of Computational Astrophysics, Chinese Academy of Sciences, 20A Datun Rd, Chaoyang District, 100012 Beijing, China*
[4]*Astronomisches Rechen-Institut, Zentrum für Astronomie, University of Heidelberg, Mönchhofstrasse 12-14, D-69120 Heidelberg, Germany*
[5]*Institute of Astronomy, University of Cambridge, Cambridge, CB3 0HA, UK*
[6]*Nicolaus Copernicus Astronomical Centre, Polish Academy of Sciences, ul. Bartycka 18, PL-00-716 Warsaw, Poland*
[7]*Main Astronomical Observatory, National Academy of Sciences of Ukraine, 27 Akademika Zabolotnoho St, UA-03680 Kyiv, Ukraine*
[8]*Max-Planck Institut für Astrophysik, Karl-Schwarzschild-Str. 1, D-85741 Garching, Germany*

NBODY6++GPU
(Wang 2015)

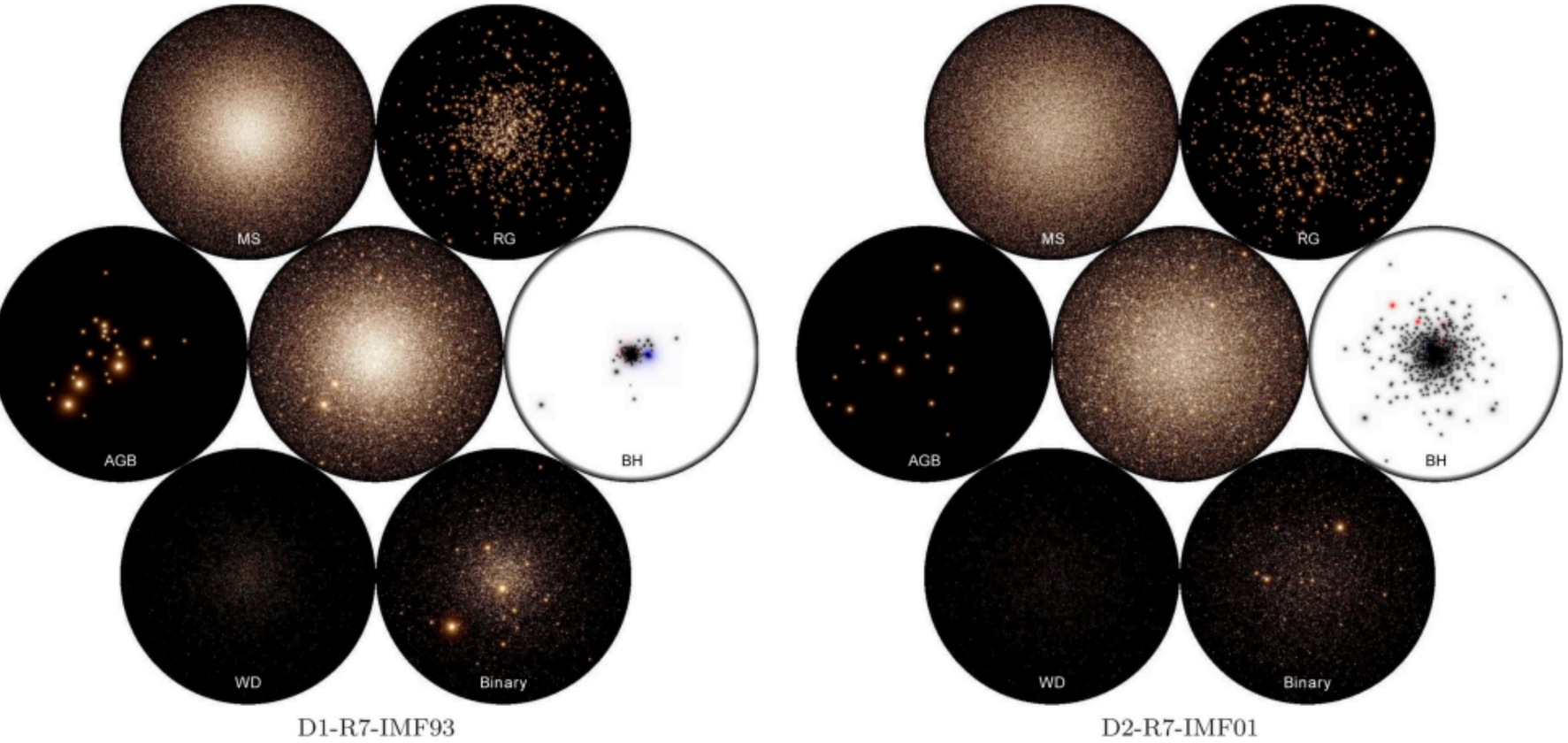


- Wang et al. (2016):
  - $N = 10^6$;
  - Initial half-mass radius: $\sim 7\ pc$
  - Primordial binary 5%
  - 160 CPU cores + 16 K20x GPUs



Hydra GPU Cluster, Garching. Germany

Dragon Simulations: Direct *N*-body with million stars



Observations of luminous stars simulated with the COCOA code (Askar et al. 2018)

- Wang et al. (2016):
  - $N = 10^6$;
  - Initial half-mass radius: $\sim 7\ pc$
  - Primordial binary 5%
  - 160 CPU cores + 16 K20x GPUs

More black holes → puffy cluster with low central surface brightness → large core and half-light radii

Other *N*-body codes for collisional *N*-body dynamics

- φ-GRAPE (Harfst et al.2007) and ϕ-GPU (Berczik et al. 2011)
  - Developed by Peter Berczik: https://github.com/berczik/phi-GPU-mole
    - Hierarchical block time-steps
    - high (4th, 6th and 8th) order Hermite integration schemes
    - Softening

- FROST & BIFROST (Rantala et al. 2023)
  - fourth-order forward symplectic integrator
  - GPU accelerated
  - High binary fraction: secular and regularised integration for binaries, triples, multiples, few body systems
  - Small softening values

- Hybrid *N*-body codes: BRIDGE (Fujii et al. 2007), PeTar (Wang et al. 2020)

N-body simulations for collisionless systems

- Softening
  - What is a reasonable value of epsilon?

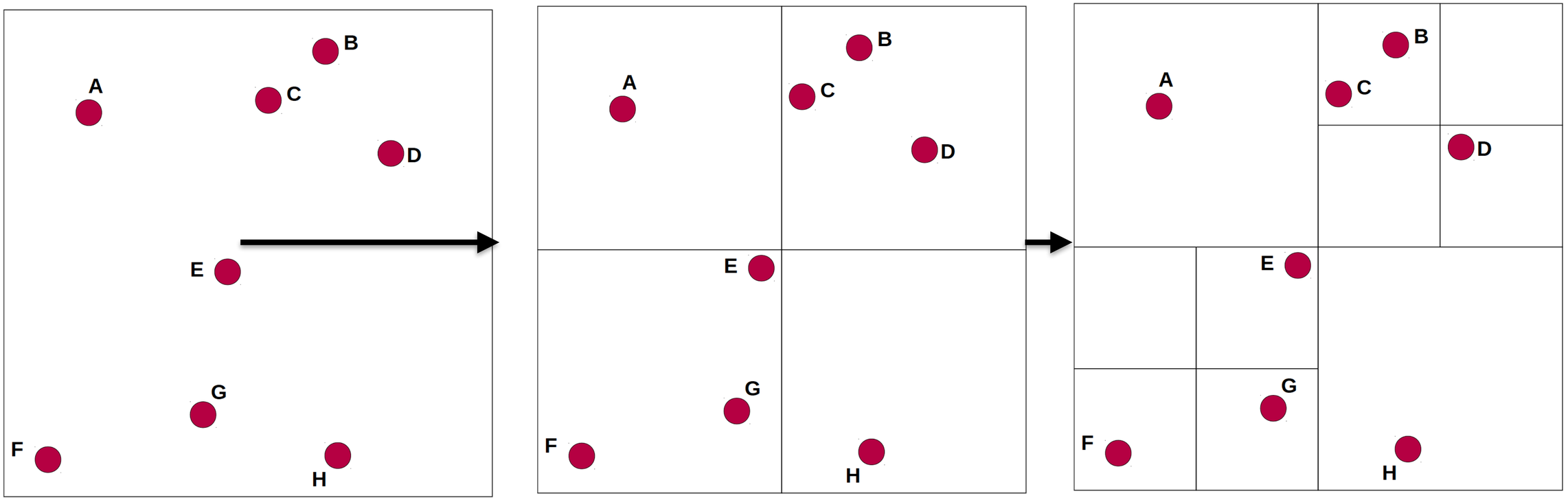
$$F_{ij} = -Gm_i m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{\left(\left| \mathbf{r}_i - \mathbf{r}_j \right|^2 + \epsilon^2 \right)^{3/2}}$$

  - Average distance between particles or a bit less
  - Adaptive individual softening
- Time steps can be large depending on the problem at hand: lower order integrators like leapfrog can be used
  - Same time-step for all particles commonly used
- Force computation: Approximating direct summation
  - No need to estimate the force exerted by each single particle on another
  - Can estimate the cumulative force exerted by groups of particles if they are sufficiently far

Tree code: Barnes-Hut Algorithm

- The crucial idea in speeding up the brute force n-body algorithm is to group nearby bodies and approximate them as a single body.
- Uses a hierarchical spatial tree to define localised groups of particles

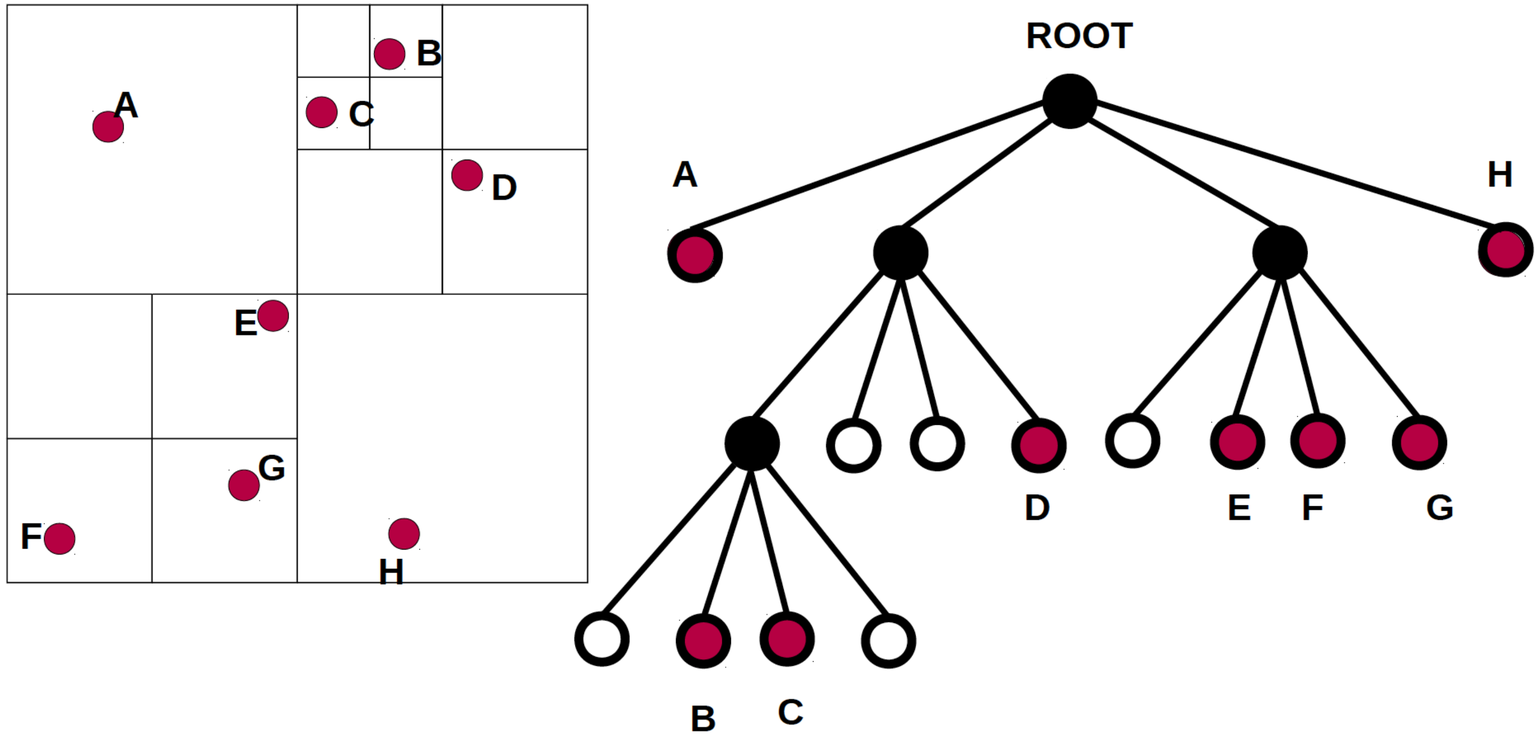
Example: quad-tree with 8 particles



Credit: Mapelli lectures on *N*-body techniques in astrophysics

Tree code: Barnes-Hut Algorithm

Example: quad-tree with 8 particles



Father node: >1 particle

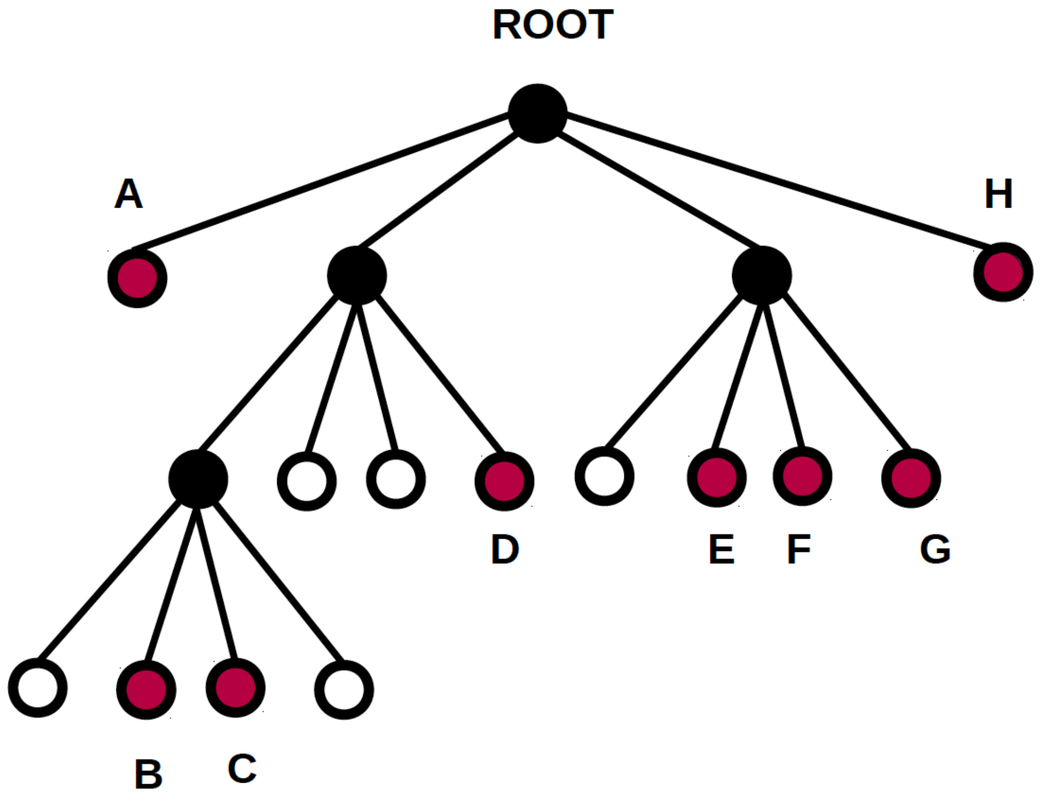
Empty daughter node

Daughter node with 1 particle

Credit: Mapelli lectures on *N*-body techniques in astrophysics

Tree code: Barnes-Hut Algorithm

- Force between nearby particles is calculated directly
  - e.g. the force exerted by F on E
- Force between distant particles is approximated with centre of mass
  - e.g. the force of E, F, G on D is approximated by the centre of mass of E, F and G
- Can significantly reduce the number of force calculations
- Need a criteria to decide when particle is far enough to avoid direct calculation
  - Based on the ratio of the width of the region represented by the considered node (s) and the distance between the body and the node's center-of-mass



$s/d < \theta$ $\theta =$ threshold value

$\theta = 0.5$ commonly used in practice

Credit: Mapelli lectures on *N*-body techniques in astrophysics

Tree code: Barnes-Hut Algorithm

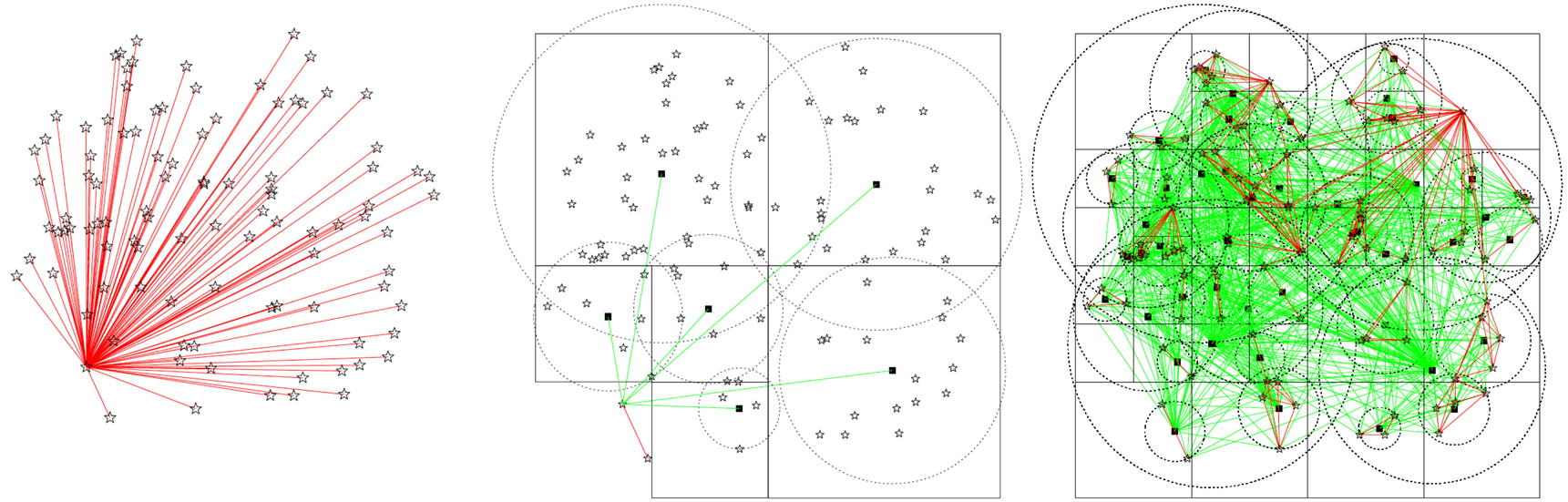


Fig. 5. Left: computation of the force for one of 100 particles (asterisks) in two dimensions (for graphical simplicity) using direct summation: every line corresponds to a single particle-particle force calculation. Middle: approximate calculation of the force for the same particle using the tree code. Cells opened are shown as black squares with their centres $\boldsymbol{z}$ indicated by solid squares and their sizes w by dotted circles. Every green line corresponds to a cell-particle interaction. Right: approximate calculation of the force for all 100 particles using the tree code, requiring 902 cell-particle and 306 particle-particle interactions ($\theta = 1$ and $n_{\max} = 1$), instead of 4950 particle-particle interactions with direct summation.

From Dehnen & Read (2011)

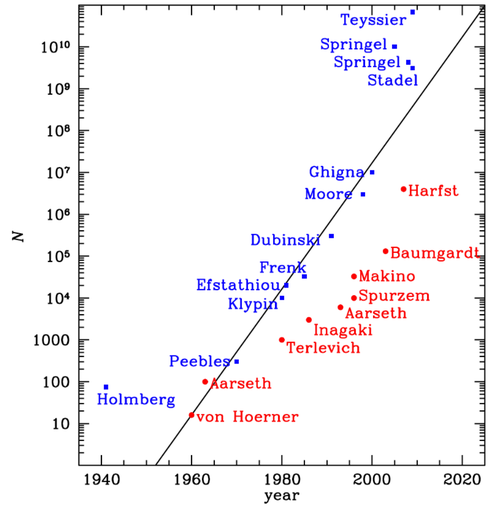
In direct summation for acceleration calculation:

$N(N-1)/2$

For N=100, 4950 particle-particle interactions

With tree code: 902 cell-particle and 306 particle-particle interactions

Evaluation of force scales as $O(N \log N)$



See Dehnen & Read (2011) for Fast multipole method (FMM) and Grid-based methods

Hybrid *N*-body code: PeTar

PeTar is an N-body code specifically designed for modeling collisional stellar systems, where factors such as multiplicity (binaries, triples, etc.) and close encounters play a crucial role in dynamical evolution. PeTar offers several key advantages:

- **Precise gravitational force modeling**: PeTar does not employ any softening of gravitational force, enabling accurate tracking of the orbital evolution of binaries, triples, and close encounters.

https://github.com/lwang-astro/PeTar

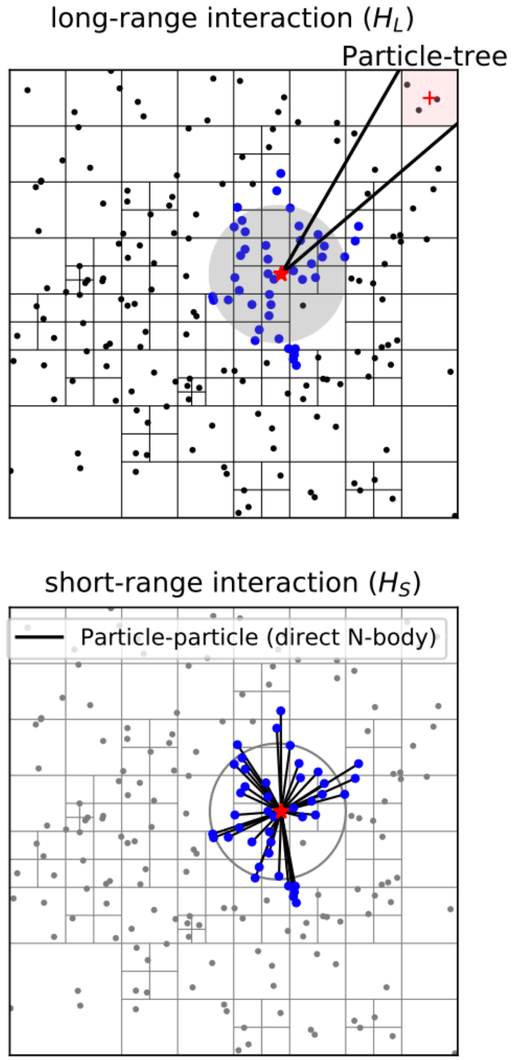


Fig. 1 from Wang et al. (2020)

- PeTar (Wang et al. 2020): Barnes-Hut Tree Algorithm + Hermite Integrator + slow-down algorithmic regularization
- Leverages multi-CPU processors/threads and GPU acceleration to accelerate simulations
  - Up to 11 times faster than NBODY6++GPU
- Can treat high multiplicity fraction (binaries and triples)
  - No Softening

Key takeaways

- Direct summation *N*-body codes for collisional systems are slow but accurate!
  - Solving equation of motions seems straightforward but computationally challenging $\sim O(N^2)$
  - Several techniques to speed up calculations:
    - Neighbour scheme
    - Hierarchical block time steps
    - Regularization
    - Parallelization
  - Hardware acceleration has enabled long-term simulations of realistic globular clusters models
    - GPUs (game changer)
- Collisionless *N*-body systems are relatively less tedious
  - Softening and lower order integrators
  - Direct summation can be avoided: Tree Methods
- Up next: Monte Carlo modelling is a realistic practical alternative to simulating large globular clusters