



The n-body problem (1/2)

Prof. Richard Vuduc

Georgia Institute of Technology

CSE/CS 8803 PNA: Parallel Numerical Algorithms

[L.20] Tuesday, March 25, 2008

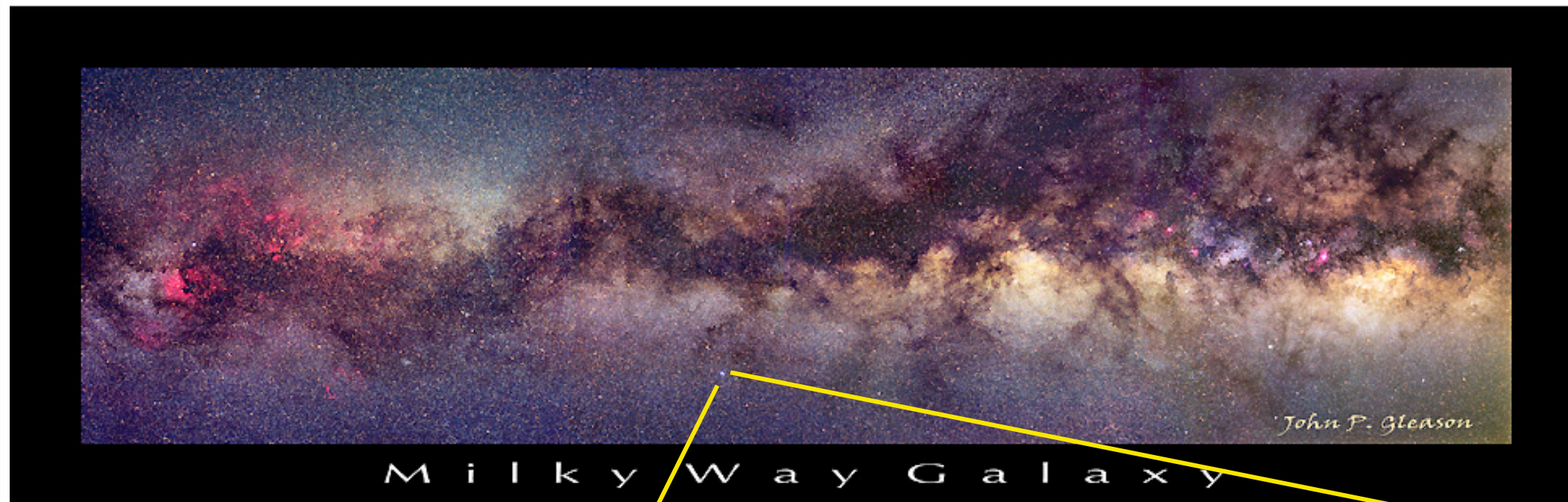


Today's sources

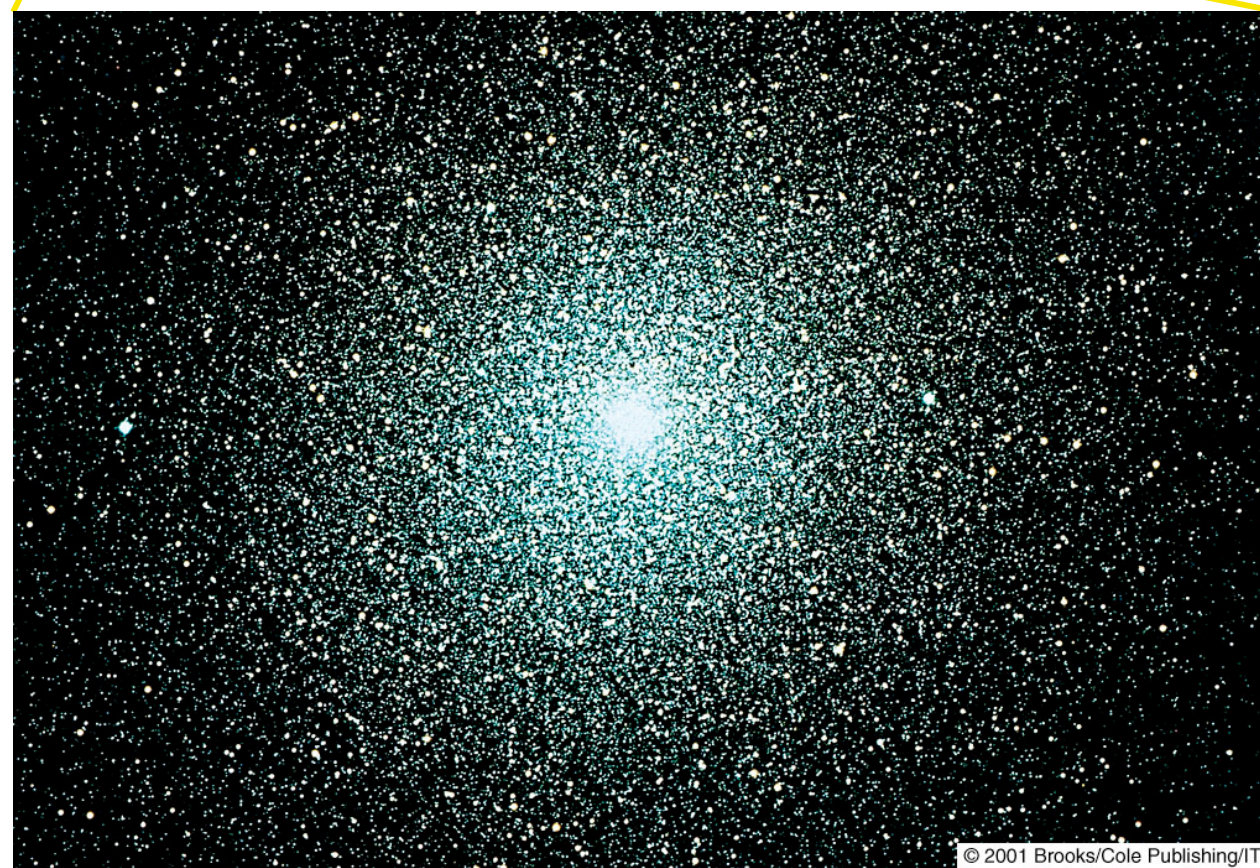
- ❑ CS 267 at UCB (Demmel & Yelick)
- ❑ *Computational Science and Engineering*, by Gilbert Strang
- ❑ Lectures 18–21 of M. Ricotti's Astro 415 course at U. Maryland
- ❑ Jim Stone (Princeton)
- ❑ Andrey Kravtsov (U. Chicago)
- ❑ Mike Heath (UIUC)
- ❑ Changa (ChaNGa, U. Washington; based on CHARM++)



Motivating example from
computational astrophysics



**Example: Stellar dynamics
in a globular cluster**



Example: Stellar dynamics in a globular cluster



Source: Frank Summers, AMNH, via Andrey Kravtsov



A little history of the n-body problem...

[1941APJ...94..385H]

THE ASTROPHYSICAL JOURNAL

AN INTERNATIONAL REVIEW OF SPECTROSCOPY AND
ASTRONOMICAL PHYSICS

VOLUME 94

NOVEMBER 1941

NUMBER 3

ON THE CLUSTERING TENDENCIES AMONG THE NEBULAE
II. A STUDY OF ENCOUNTERS BETWEEN LABORATORY MODELS OF
STELLAR SYSTEMS BY A NEW INTEGRATION PROCEDURE

ERIK HOLMBERG



Systems innovation! Who needs computers?

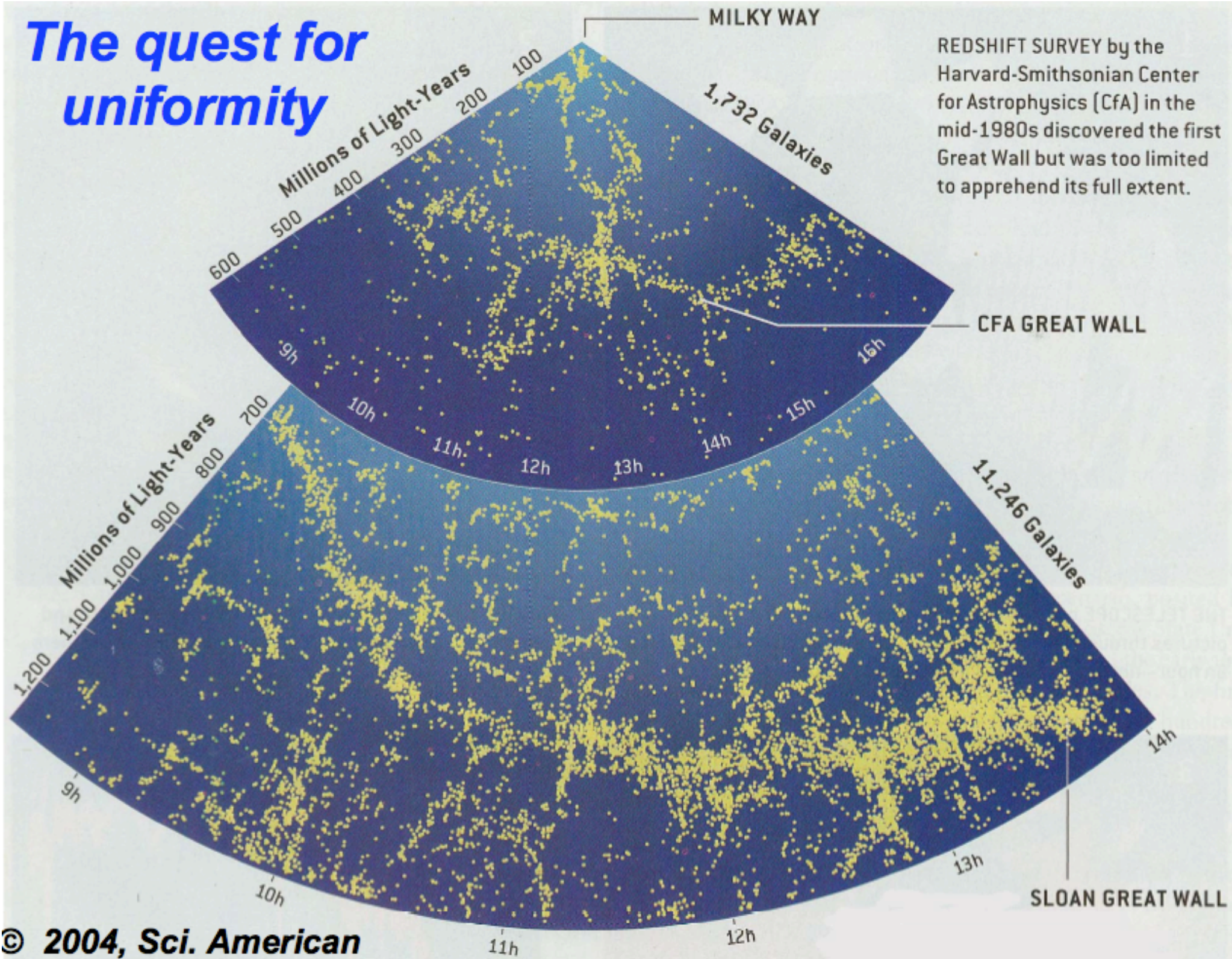
386

ERIK HOLMBERG

a certain mass element belonging to one of the two nebulae, we must first derive as a function of the time the x and y components of the total gravitational force acting upon the element. Starting from a certain distribution of mass in the nebulae, we may find the total gravitation by a purely numerical integration. However, such an integration is impracticable on account of the large amount of work involved. In the present case a solution has been found by replacing gravitation by light. Every mass element is represented by a small light-bulb, the light being proportional to the mass, and the total light along the x and y axes is measured by a combination of a photocell and a galvanometer. The measured values represent the components of the gravitational force. The latter components are obtained by adding up the attractions due to individual mass elements, each multiplied by the cosine of the corresponding projection angle. Consequently, the photocell must obey the cosine law as far as the angle of incidence of the light is concerned. If the photocell obeys the cosine law and if the combination of photocell and galvanometer gives a linear relation between light and scale reading, the galvanometer deflection will be proportional to the total gravitational force or, more correctly, to the total acceleration. A detailed account of the instrumental arrangement is given below.

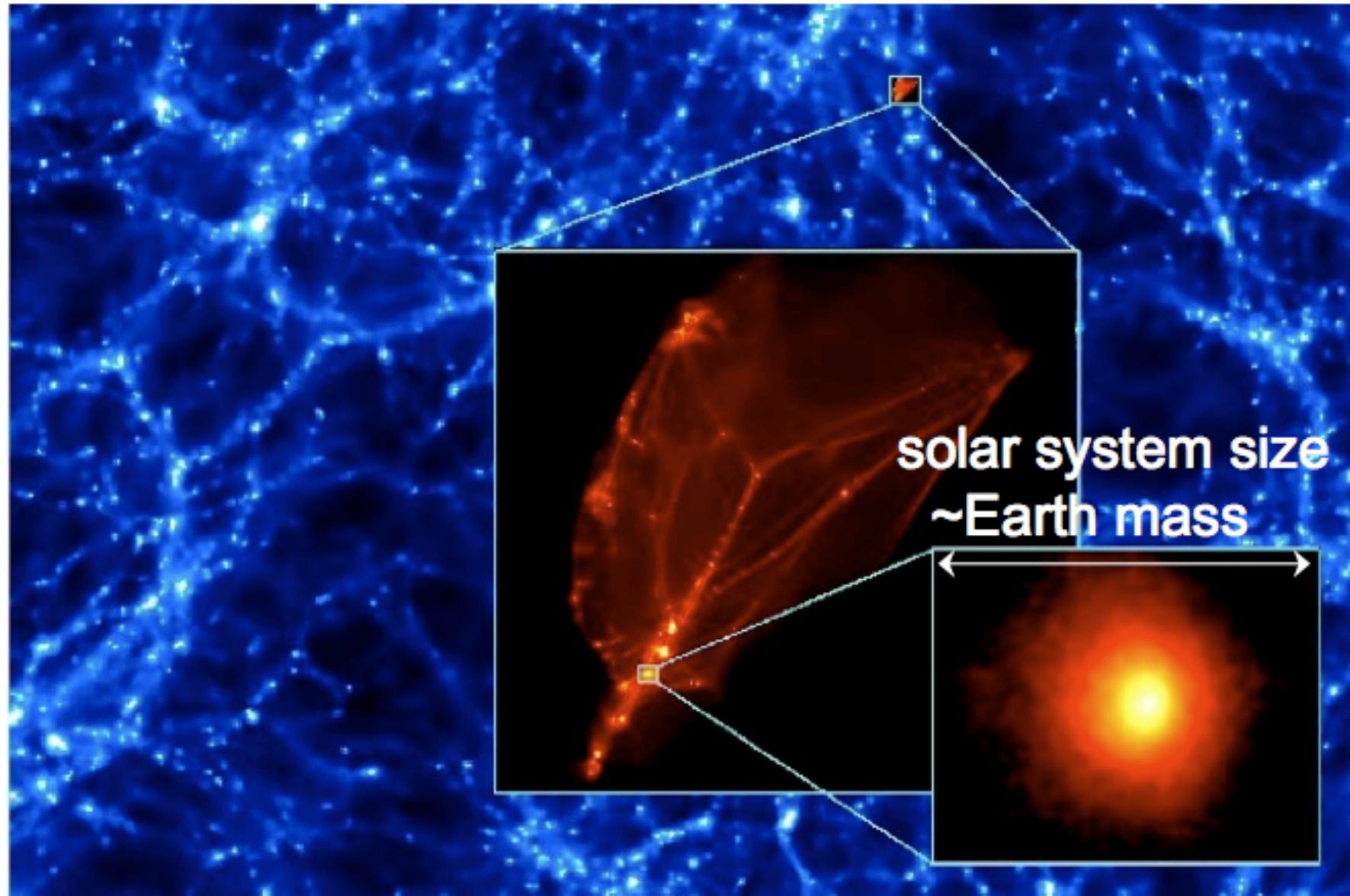
The light-bulbs, which represent the individual mass elements, must fulfil three different requirements. Their candle powers must be the same within certain limits. Furthermore, the candle power must not change when the light-bulb is rotated about

The quest for uniformity



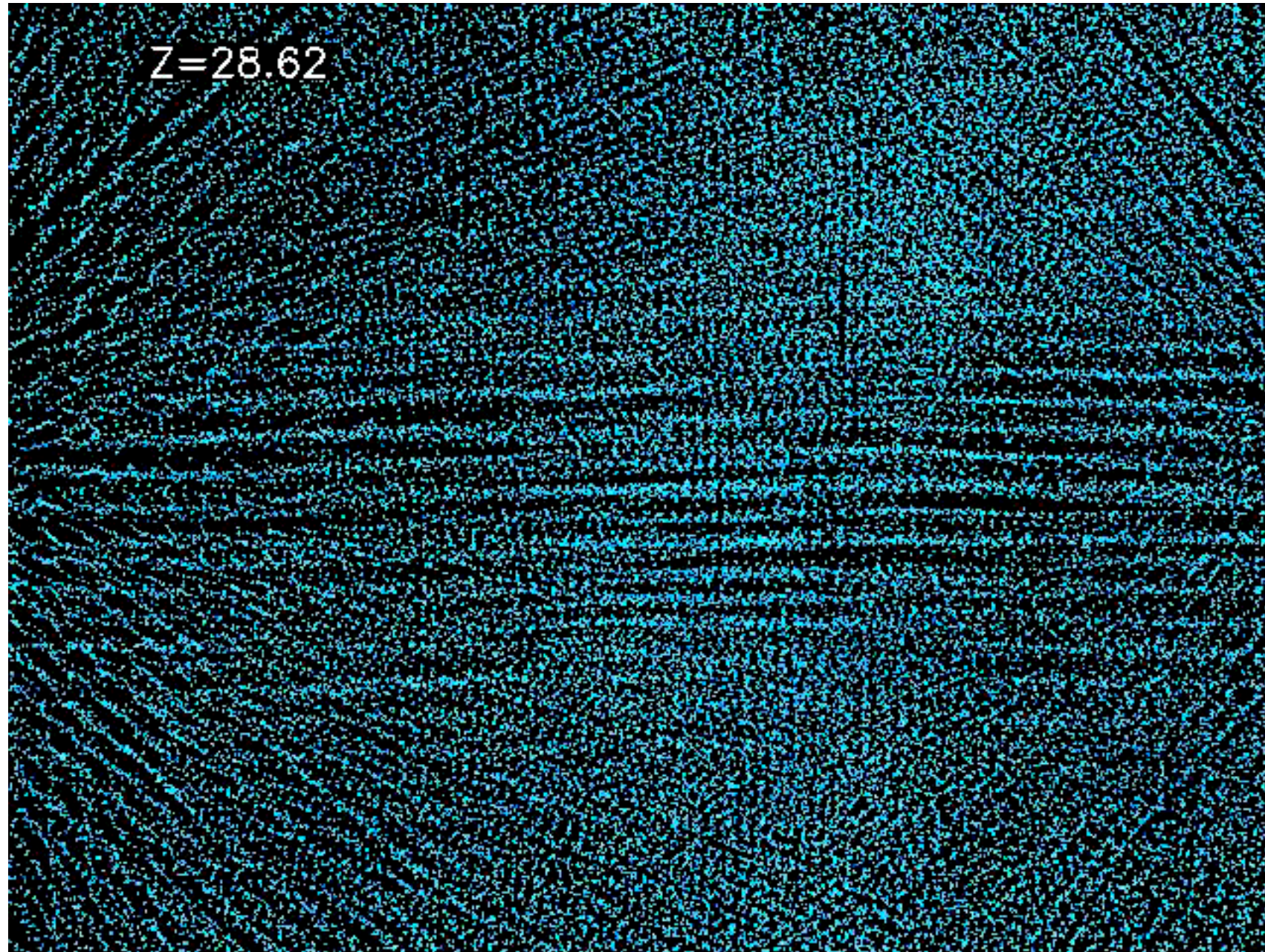
Example: Formation of group of galaxies

100 million light years $\approx 10^{20}$ km



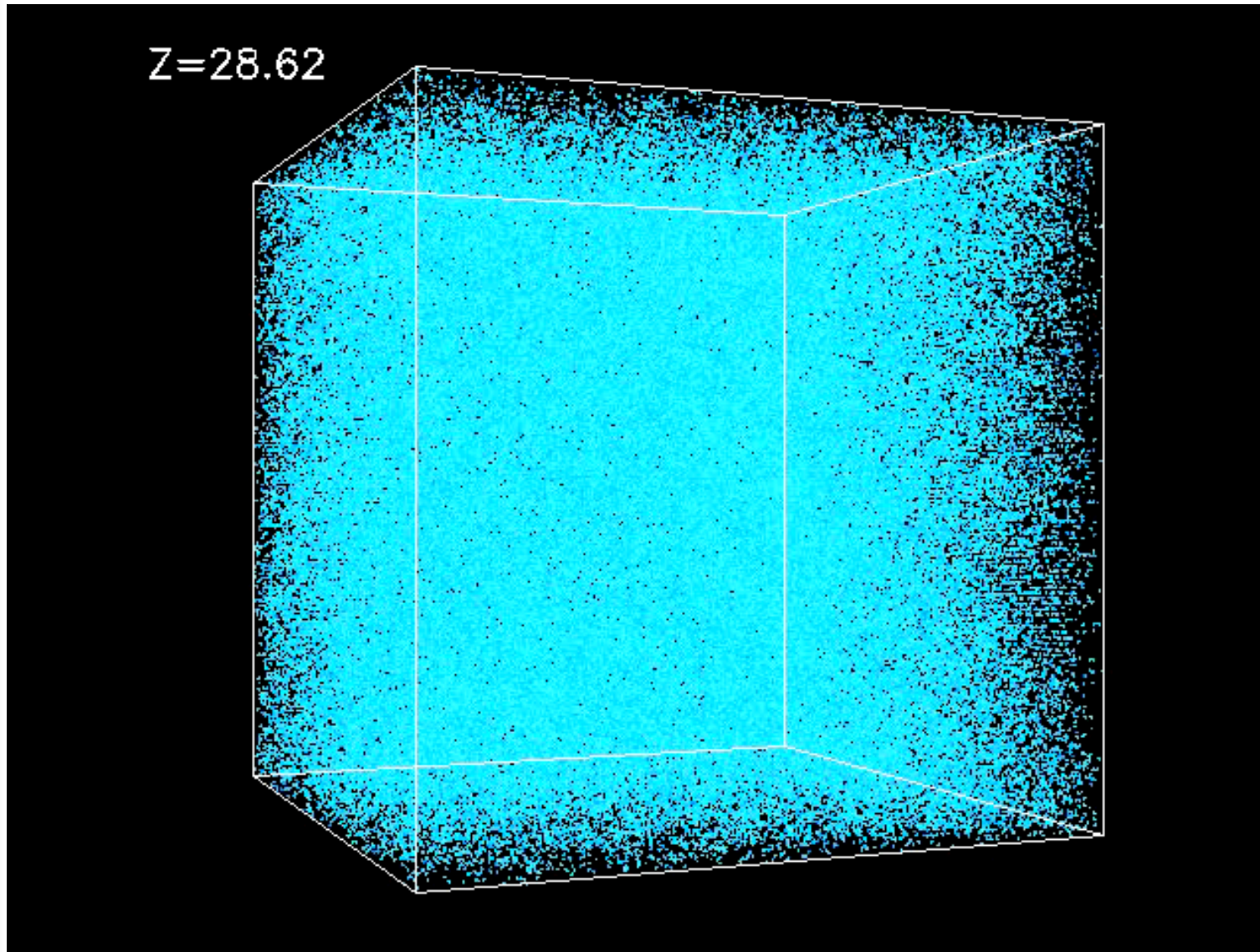
Source: Ben Moore, U. Zurich (via Andrey Kravtsov) – <http://nbody.net>

Example: Formation of group of galaxies



Source: Andrey Kravtsov, U. Chicago – <http://cosmicweb.uchicago.edu>

Example: Dark matter filament formation



- Particles = dark matter
- Dim: 43 Mpc $\approx 10^{21}$ km

Source: Andrey Kravtsov, U. Chicago – <http://cosmicweb.uchicago.edu>



Length-/time-scale challenges

- Long length-scales
 - Star cluster is 10^9 times larger than star
 - Neutron star: 10^{14} x
- Long time-scales
 - Close passage between stars ~ hours
 - Evolution of cluster ~ 10^9 years $\Rightarrow 10^{14}$ x (neutron stars: 10^{20})

Newton's laws of motion and gravity

$$\begin{aligned}\mathbf{F}_i &= m_i \frac{d^2 \mathbf{r}_i(t)}{dt^2} \\ &\equiv m_i \ddot{\mathbf{r}}_i \\ &= -G m_i \sum_{j \neq i} m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^3} \\ &\Downarrow \\ \ddot{\mathbf{r}}_i &= -G \sum_{j \neq i} m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^3}\end{aligned}$$

Rewrite as 1st-order ODE

$$\frac{\mathbf{F}_i}{m_i} = \ddot{\mathbf{r}}_i = -G \sum_{j \neq i} m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^3}$$
$$\Downarrow$$
$$\frac{d}{dt} \begin{pmatrix} \mathbf{r}_i \\ \mathbf{v}_i \end{pmatrix} = \begin{pmatrix} \mathbf{v}_i \\ \mathbf{F}_i/m_i \end{pmatrix}$$

n particles $\Rightarrow 6n$ -element vector



Computational astrophysics

- Sample problems
 - Planetary orbits over billions of years
 - Planetary, star, and galaxy formation
 - Star cluster evolution over time
- Core computation: **N-body problem**
 - **Integrate ODE** – Off-the-shelf methods
 - **Inter-particle force evaluation** – $O(n^2)$ → $O(n \log n)$ → $O(n)$ algorithms
- Applies to other domains, *e.g.*, molecular dynamics, ...

Progress in N-body simulations for globular clusters

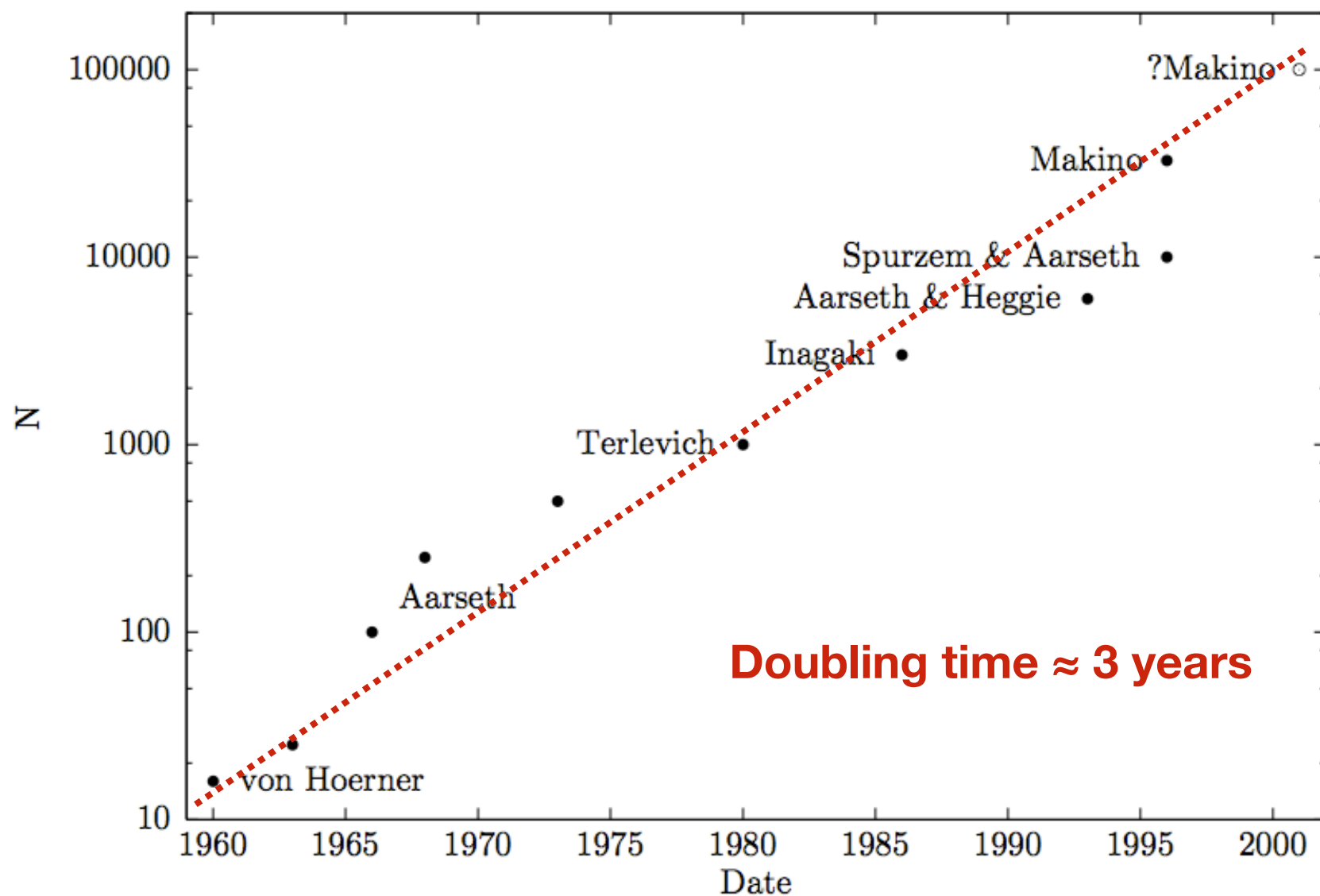


Fig. 1. The slow progress of N -body simulations of star clusters. Models computed well into the late evolution are plotted against publication date. For a human perspective, see Aarseth (1999) and von Hoerner (2001).

Source: James Stone (Princeton)



ODE solvers



Recall: Computational tasks

- Setup
- Solve initial-value problem (ODE)
 - Evaluate forces
 - Integrate over some time step

Setup initial distribution of particles



Source: Jim Stone (Princeton)

Solve initial-value problem (ODE), with force evaluation at each step

$$\frac{d}{dt} \begin{pmatrix} \mathbf{r}_i \\ \mathbf{v}_i \end{pmatrix} = \begin{pmatrix} \mathbf{v}_i \\ \mathbf{F}_i/m_i \end{pmatrix}$$

\Downarrow

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



“Soften” to remove singularities

$$\frac{d}{dt} \begin{pmatrix} \mathbf{r}_i \\ \mathbf{v}_i \end{pmatrix} = \begin{pmatrix} \mathbf{v}_i \\ \mathbf{F}_i/m_i \end{pmatrix}$$
$$\Downarrow$$
$$\frac{\mathbf{F}_i}{m_i} = \ddot{\mathbf{r}}_i = -G \sum_{j \neq i} m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{(|\mathbf{r}_i - \mathbf{r}_j|^2 + \epsilon^2)^{3/2}}$$



IVP ODE solvers

■ Many algorithms

- Taylor series (e.g., Euler)
- Runge-Kutta
- Extrapolation
- Multistep
- Multivalued

■ Design space

- Single vs. multistep
- Explicit vs. implicit
- No. of req'd function/derivative evaluations



IVP ODE solvers

- Many algorithms

- **Taylor series** (*e.g.*, Euler)

- Runge-Kutta

- Extrapolation

- Multistep

- Multivalued

- Design space

- Single vs. multistep

- Explicit vs. implicit

- No. of req'd function/
derivative evaluations



Taylor series-based methods

- Derive finite-difference approximations using Taylor series

$$\text{ODE: } \dot{y} = f(t, y(t))$$

$$y(t+h) = y(t) + h\dot{y}(t) + \frac{h^2}{2}\ddot{y}(t) + \dots$$

- Example: Forward Euler

$$t_k = t_0 + h \cdot k \quad k \in \{1, 2, \dots\}$$

$$y_{k+1} \leftarrow y_k + h \cdot f(t_k, y_k)$$

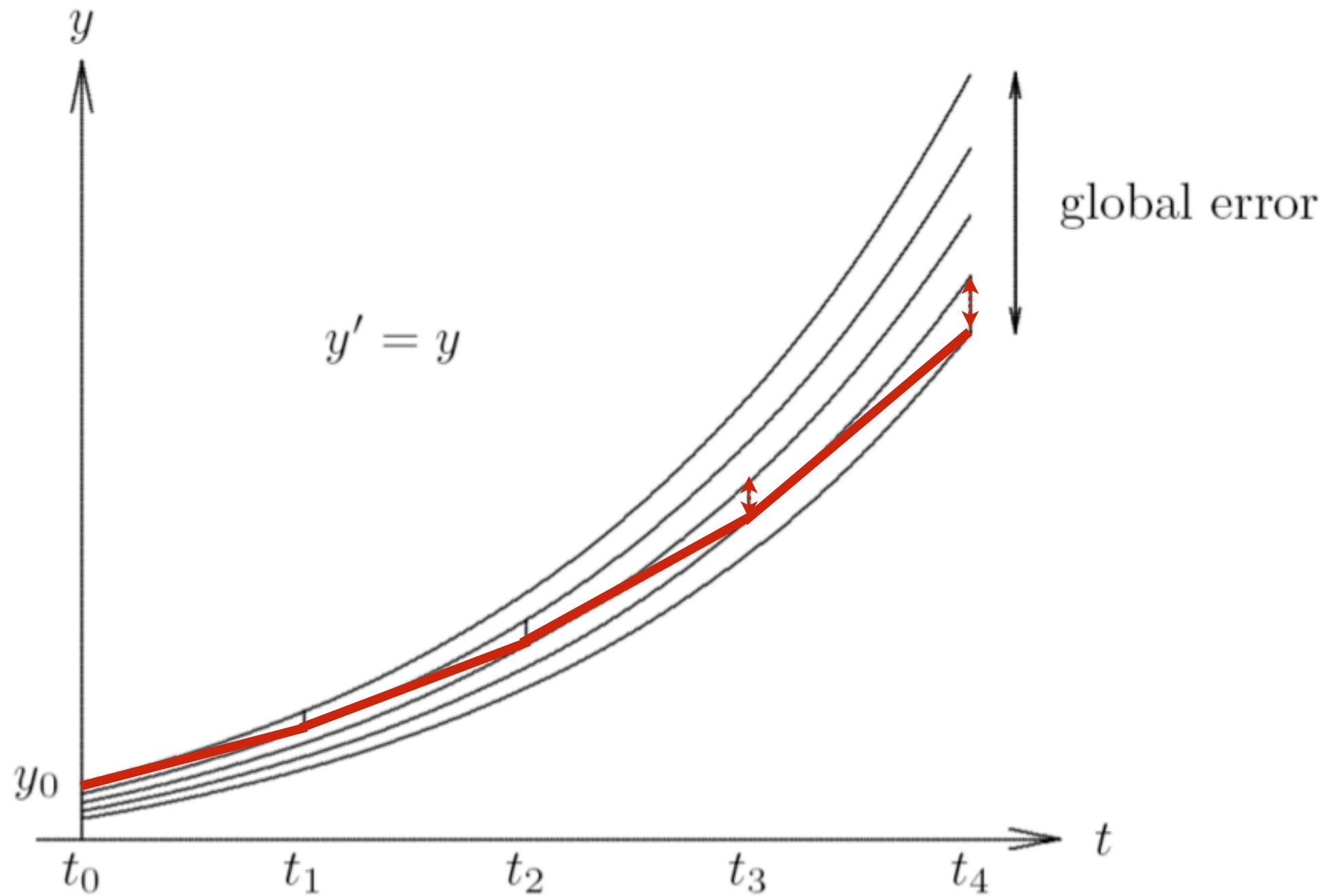


Taylor series-based methods: Forward Euler

$$t_k = t_0 + h \cdot k \quad k \in \{1, 2, \dots\}$$
$$y_{k+1} \leftarrow y_k + h \cdot f(t_k, y_k)$$

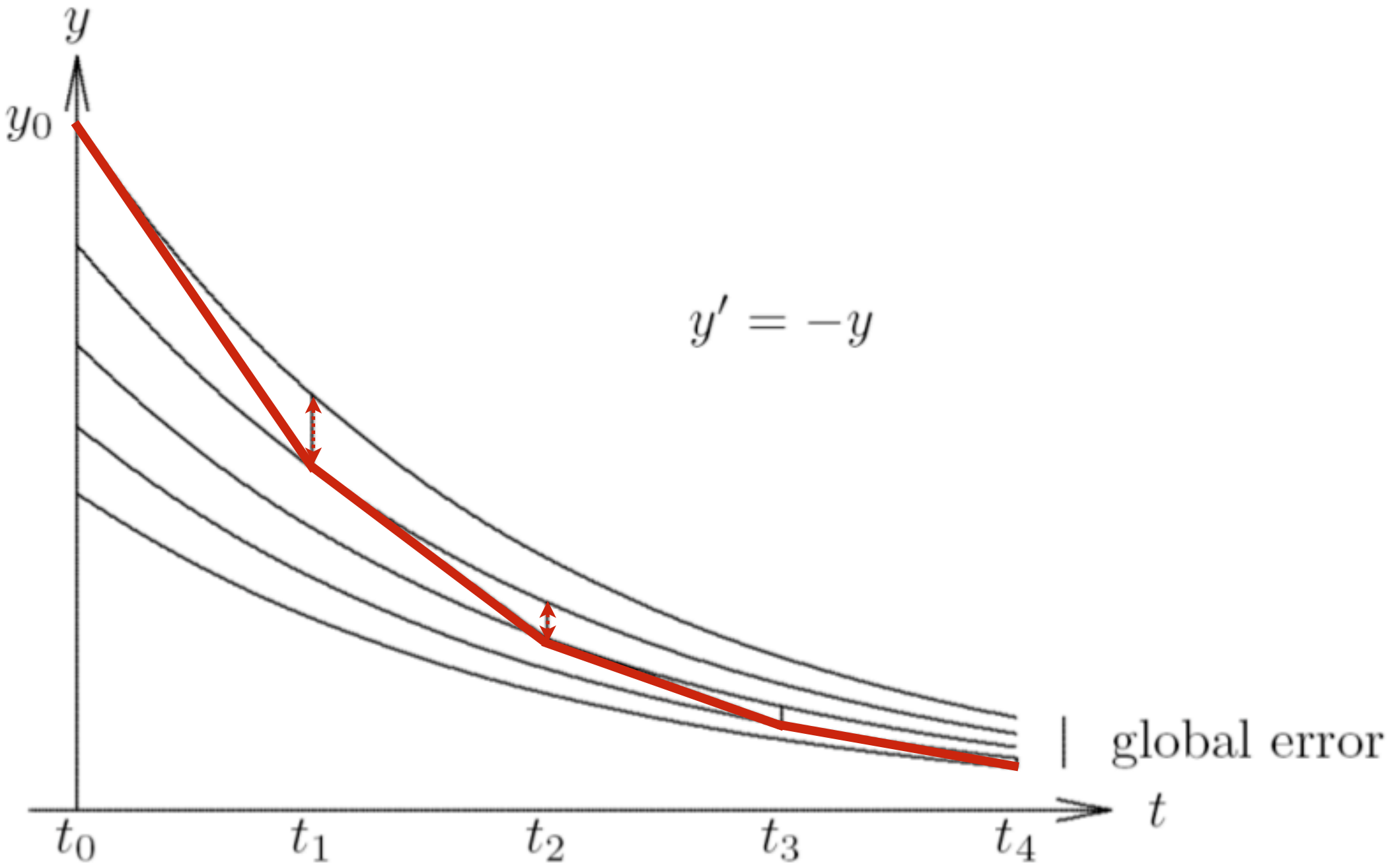
- **1st-order accurate** (Higher order: Take more derivative terms)
- **Single-step** (variably-sized steps possible)
- **Explicit:** y_{k+1} depends only on t_k
- Narrow stability window, *i.e.*, small range of values for h for stability

Forward Euler: Local and global errors




Source: M. Heath (UIUC)

Forward Euler: Local and global errors



Source: M. Heath (UIUC)





Taylor series-based methods: Backward Euler

$$y(t - h) = y(t) - h\dot{y}(t) + \frac{h^2}{2}\ddot{y}(t) + \dots$$
$$y_k \leftarrow y_{k-1} + h \cdot f(t_k, y_k)$$

- **Implicit:** y_{k+1} depends only on t_k
- **Larger stability window** vs. forward Euler
- 1st-order accurate (Higher order: Take more derivative terms)
- Single-step (variably-sized steps possible)



IVP ODE solvers

- Many algorithms

- Taylor series (e.g., Euler)

- **Runge-Kutta**

- Extrapolation

- Multistep

- Multivalued

- Design space

- Single vs. multistep

- Explicit vs. implicit

- No. of req'd function/
derivative evaluations



Runge-Kutta methods

- Single-step, but w/ high-order derivs replaced by FD approx in $[t_k, t_{k+1}]$
- Classical 4th-order RK:

$$y_{k+1} \leftarrow y_k + \frac{h_k}{6} (k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = f(t_k, y_k)$$

$$k_2 = f\left(t_k + \frac{h_k}{2}, y_k + \frac{h_k}{2}k_1\right)$$

$$k_3 = f\left(t_k + \frac{h_k}{2}, y_k + \frac{h_k}{2}k_2\right)$$

$$k_4 = f(t_k + h_k, y_k + h_k k_3)$$



Runge-Kutta methods

- Nice properties
 - Self-starting
 - Easy to program
- Possible downsides
 - No a prior guidance on step-size selection
 - Several function evaluations
- “Workarounds:” Embedded RK methods; automation; implicit versions



IVP ODE solvers

- Many algorithms

- Taylor series (e.g., Euler)

- Runge-Kutta

- **Extrapolation**

- Multistep

- Multivalued

- Design space

- Single vs. multistep

- Explicit vs. implicit

- No. of req'd function/
derivative evaluations



IVP ODE solvers

- Many algorithms

- Taylor series (e.g., Euler)

- Runge-Kutta

- Extrapolation

- **Multistep**

- Multivalued

- Design space

- Single vs. multistep

- Explicit vs. implicit

- No. of req'd function/
derivative evaluations



Multistep methods

- “Memory:” Use information at more than one prior time step
- General form of *linear* multistep methods:

$$y_{k+1} \leftarrow \sum_{i=1}^m \alpha_i y_{k+1-i} + h \sum_{i=0}^m \beta_i f(t_{k+1-i}, y_{k+1-i})$$

- Obtain α , β coefficients via polynomial interpolation
- $\beta_0 = 0 \Rightarrow$ explicit; otherwise, implicit
- Predictor-corrector: Explicit guesses, implicit improvements



Multistep method examples

- Simplest 2nd-order explicit two-step method

$$y_{k+1} \leftarrow y_k + \frac{h}{2}(3\dot{y}_k - \dot{y}_{k-1})$$

- Simplest 2nd-order implicit two-step method

$$y_{k+1} = y_k + \frac{h}{2}(\dot{y}_{k+1} + \dot{y}_k)$$

- 4th-order Adams-Bashforth predictor

$$y_{k+1} \leftarrow y_k + \frac{h}{24}(55\dot{y}_k - 59\dot{y}_{k-1} - 37\dot{y}_{k-2} - 9\dot{y}_{k-3})$$

- 4th-order implicit Adams-Moulton corrector

$$y_{k+1} = y_k + \frac{h}{24}(9\dot{y}_{k+1} + 19\dot{y}_k - 5\dot{y}_{k-1} + \dot{y}_{k-2})$$



Multistep methods tradeoffs

- Upsides
 - Good local error estimates via (predictor) - (corrector)
 - Interpolation-based \Rightarrow results at points other than integration points
 - Well-designed implicit multistep good for “stiff” problems
- Downsides
 - Not self-starting
 - Complicated to change step size
 - Complicated to program



IVP ODE solvers

- Many algorithms

- Taylor series (e.g., Euler)

- Runge-Kutta

- Extrapolation

- Multistep

- **Multivalued**

- Design space

- Single vs. multistep

- Explicit vs. implicit

- No. of req'd function/
derivative evaluations



Multivalued methods

- Multistep with variable (adaptive) time steps
- Example: 4-value method

$$y(t+h) = y(t) + hy'(t) + \frac{h^2}{2}y''(t) + \frac{h^3}{6}y'''(t) + \dots$$

$$Y_k \equiv \begin{bmatrix} y_k \\ hy'_k \\ h^2/2 \cdot y''_k \\ h^3/6 \cdot y'''_k \end{bmatrix}$$
$$Y_{k+1} \leftarrow \begin{bmatrix} 1 & 1 & 1 & 1 \\ & 1 & 2 & 3 \\ & & 1 & 3 \\ & & & 1 \end{bmatrix} \cdot Y_k + \alpha$$



IVP ODE solvers

■ Many algorithms

- Taylor series (e.g., Euler)
- Runge-Kutta
- Extrapolation
- Multistep
- Multivalued

■ Design space

- Single vs. multistep
- Explicit vs. implicit
- No. of req'd function/derivative evaluations



Sources of parallelism

- Multistage, *e.g.*, Runge-Kutta: Stage evaluation
- Evaluating right-hand side, *e.g.*, forces in gravitational n-body
- Solving linear/non-linear systems, *e.g.*, implicit methods
- Partitioning equations in multiple tasks – next



Administrivia



Upcoming schedule changes

- Some adjustment of topics (TBD)
- Tu 4/1 — No class
- Th 4/3 — Attend talk by Doug Post from DoD HPC Modernization Program



Parallel ODEs: Waveform relaxation



Picard iteration

- Consider IVP, a system of n-ODEs

$$\begin{aligned}\dot{\mathbf{y}} &= \mathbf{f}(t, \mathbf{y}) \\ t &\geq t_0 \\ \mathbf{y}(t_0) &= \mathbf{y}_0\end{aligned}$$

- Idea: Following iteration converges to soln. of IVP (if f is Lipschitz)

$$Y_{k+1}(t) \leftarrow \mathbf{y}_0 + \int_{t_0}^t \mathbf{f}(s, Y_k(s)) ds$$

- **n independent 1-D quadrature calculations!**

Jacobi waveform relaxation

- Decouple system into n independent ODEs
- Consider $n = 2$

$$\frac{d}{dt} \begin{pmatrix} y_1^{(k+1)} \\ y_2^{(k+1)} \end{pmatrix} \leftarrow \begin{pmatrix} f_1 \left(t, y_1^{(k+1)}, y_2^{(k)} \right) \\ f_2 \left(t, y_1^{(k)}, y_2^{(k+1)} \right) \end{pmatrix}$$

- Red-black Gauss-Seidel (and other) splittings possible



Back to the n-body problem



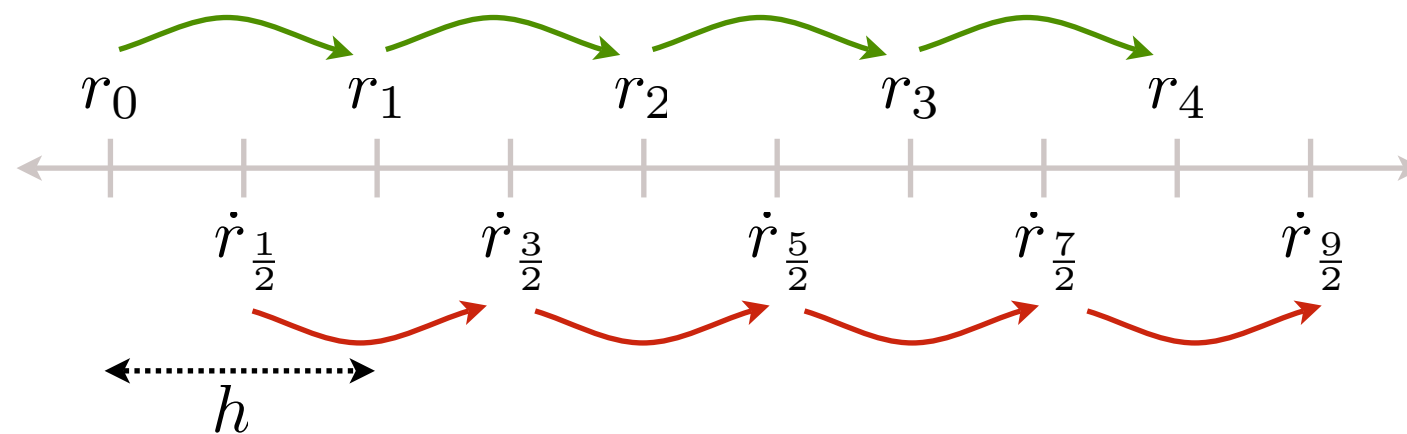
Recall: Computational tasks

- Setup
- Solve IVP
 - Evaluate forces
 - Integrate over some time step

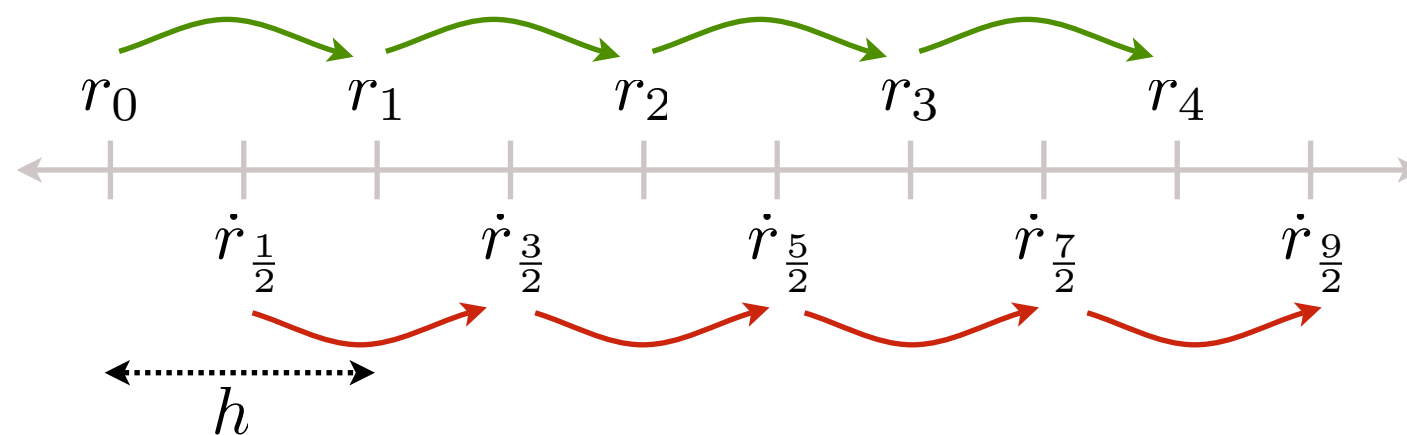
Leap-frog integrator

$$\ddot{\mathbf{r}}(t) = g(\mathbf{r}(t))$$

- Consider discretized position and velocity on a staggered grid:



Leap-frog integrator



$$v_{k+\frac{1}{2}} \equiv \dot{r} \left(t + \frac{h}{2} \right) \quad \left| \quad \begin{array}{l} r_{k+1} \leftarrow r_k + h \cdot v_{k+\frac{1}{2}} \\ v_{k+\frac{3}{2}} \leftarrow v_{k+\frac{1}{2}} + h \cdot g(r_{k+1}) \end{array} \right.$$

Time reversibility of leap-frog

$$r_{k+1} \leftarrow r_k + h \cdot v_{k+\frac{1}{2}}$$

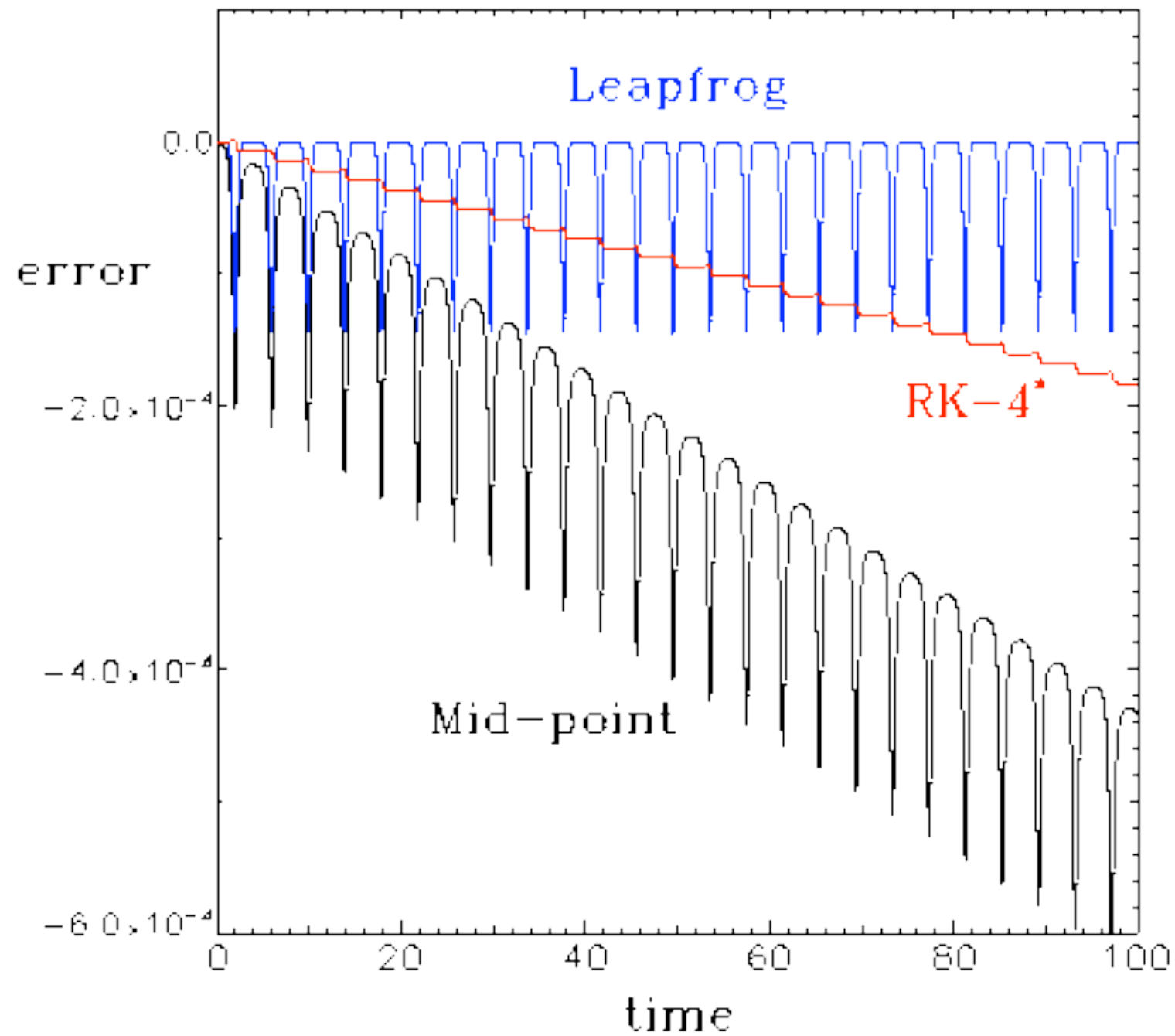
$$v_{k+\frac{3}{2}} \leftarrow v_{k+\frac{1}{2}} + h \cdot g(r_{k+1})$$

\Downarrow

$$v_{k+\frac{1}{2}} \leftarrow v_{k+\frac{3}{2}} - h \cdot g(r_{k+1})$$

$$r_k \leftarrow r_{k+1} - h \cdot v_{k+\frac{1}{2}}$$

Time reversibility of leap-frog



Source: http://www.physics.drexel.edu/courses/Comp_Phys/Integrators/leapfrog/



“In conclusion...”



Backup slides