The n-body problem (2/3)

Prof. Richard Vuduc

Georgia Institute of Technology

CSE/CS 8803 PNA: Parallel Numerical Algorithms

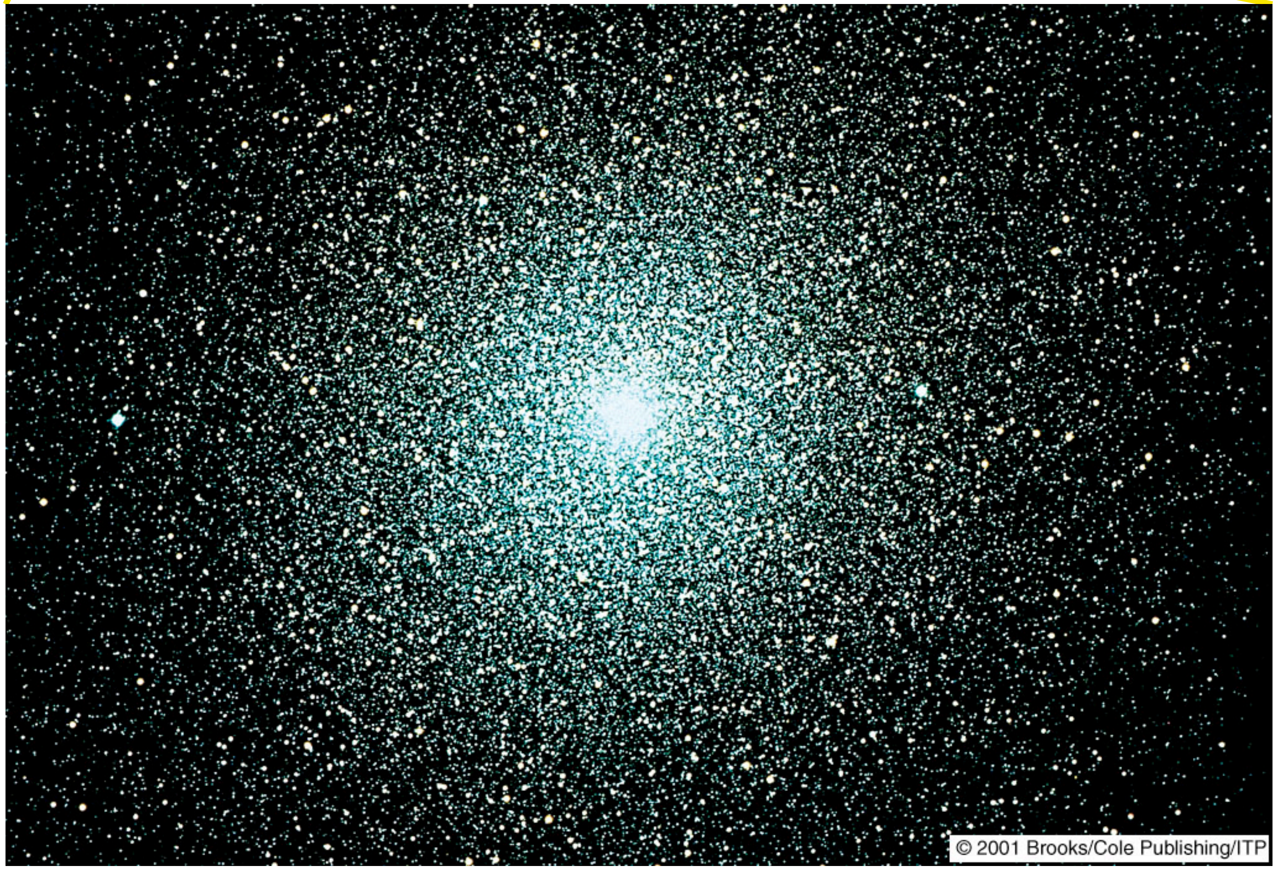
[L.21] Thursday, March 27, 2008

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++)

Review:
Parallel ODE solvers

Example: Stellar dynamics in a globular cluster



Rewrite as 1st-order ODE

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

$$\Downarrow$$

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

***n* particles ⇒ 6*n*-element vector**

Given:

$$\frac{\mathbf{F}_i}{m_i} = \ddot{\mathbf{r}}_i = -G \sum_{j \neq i} m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{\left(|\mathbf{r}_i - \mathbf{r}_j|^2 + \epsilon^2\right)^{\frac{3}{2}}}$$

$$\mathbf{r}_i(t_0), \mathbf{v}_i(t_0) = \ldots$$

Solve:

$$t \geq t_0$$

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

***n* particles ⇒ 6*n*-element vector**

Computational tasks:

1. **Solve ODE**
2. **Inner loop: compute forces**

IVP ODE solvers

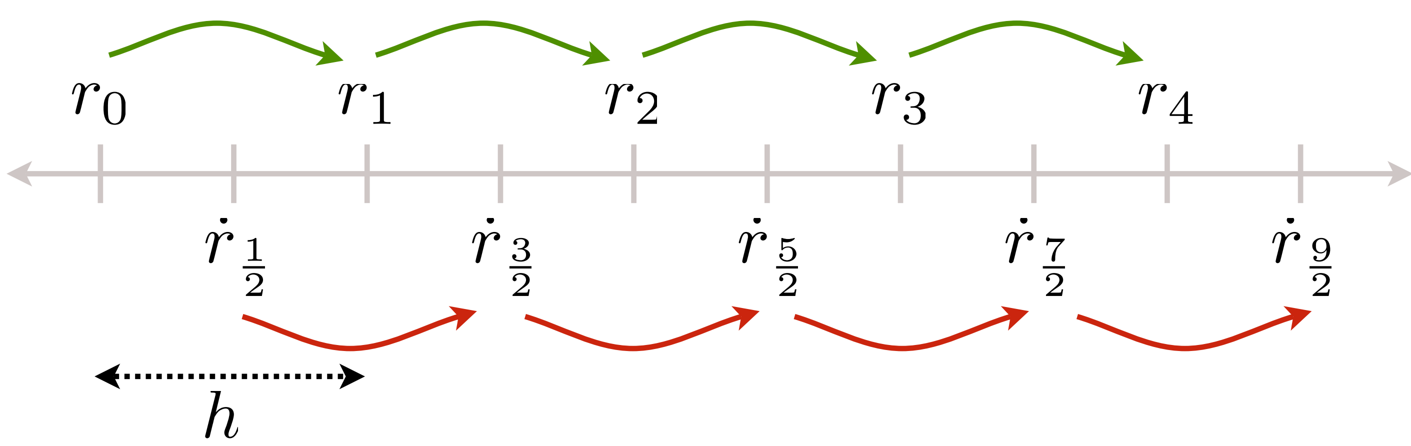
- Many algorithms
  - Taylor series (*e.g.*, Euler)
  - Runge-Kutta
  - Extrapolation
  - Multistep
  - Multivalue
- Design space
  - Single vs. multistep
  - Explicit vs. implicit
  - No. of req'd function/ derivative evaluations

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 – waveform relaxation

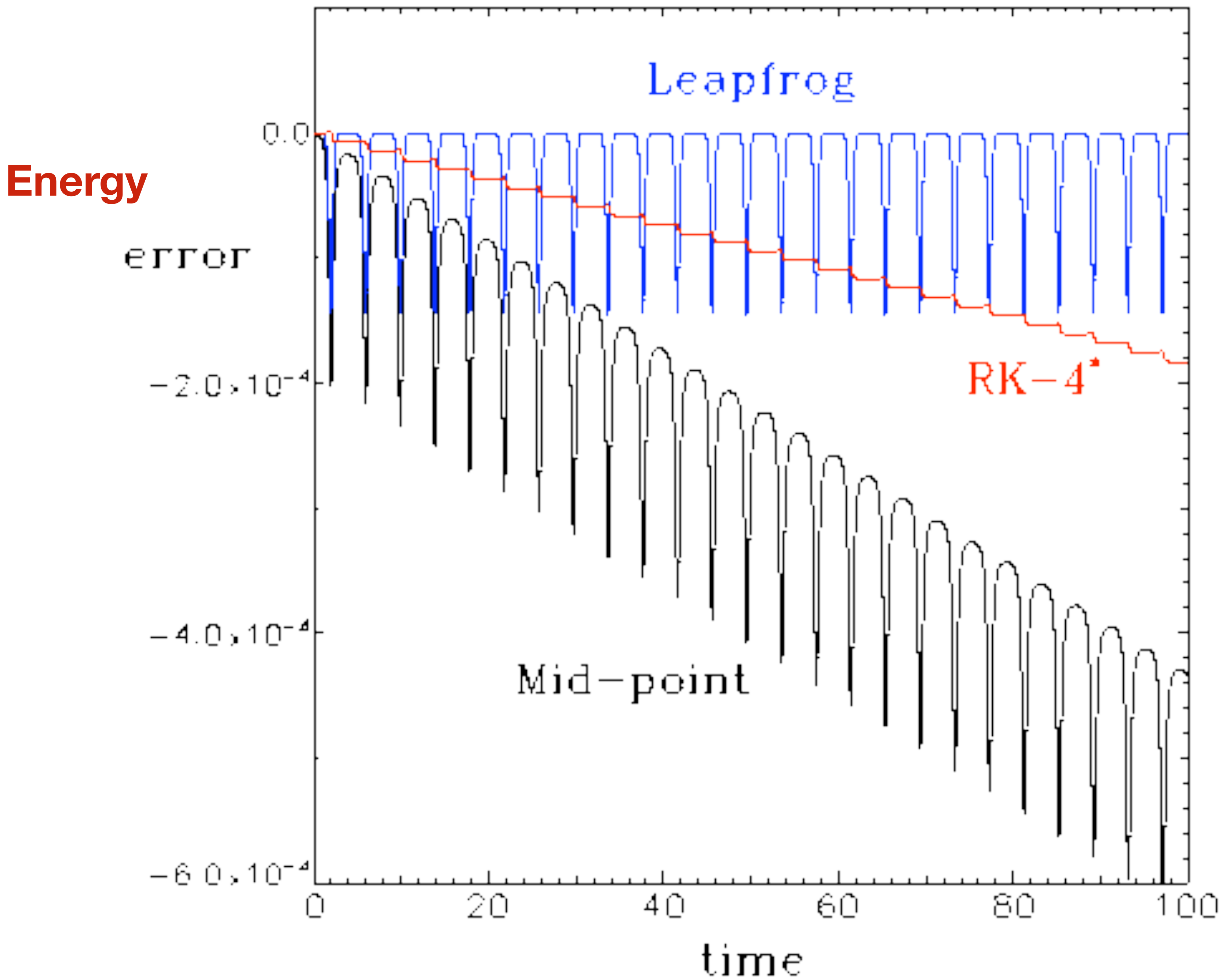
$$\frac{d}{dt}\begin{pmatrix} y_1^{(k+1)} \\ y_2^{(k+1)} \end{pmatrix} \leftarrow \begin{pmatrix} f_1\left(t, y_1^{(k+1)}, y_2^{(k)}\right) \\ f_2\left(t, y_1^{(k)}, y_2^{(k+1)}\right) \end{pmatrix}$$

Leap-frog integrator



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

Time reversibility of leap-frog



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

Efficiently computing forces: Particle-mesh (PM) approach

Method 1: Direct summation ("Particle-Particle")

- Most straightforward and accurate
- Expensive ⇒ $O(N^2)$

- Parallelization?

Method 2: Particle-Mesh

- Idea: Gravitational field has a scalar potential

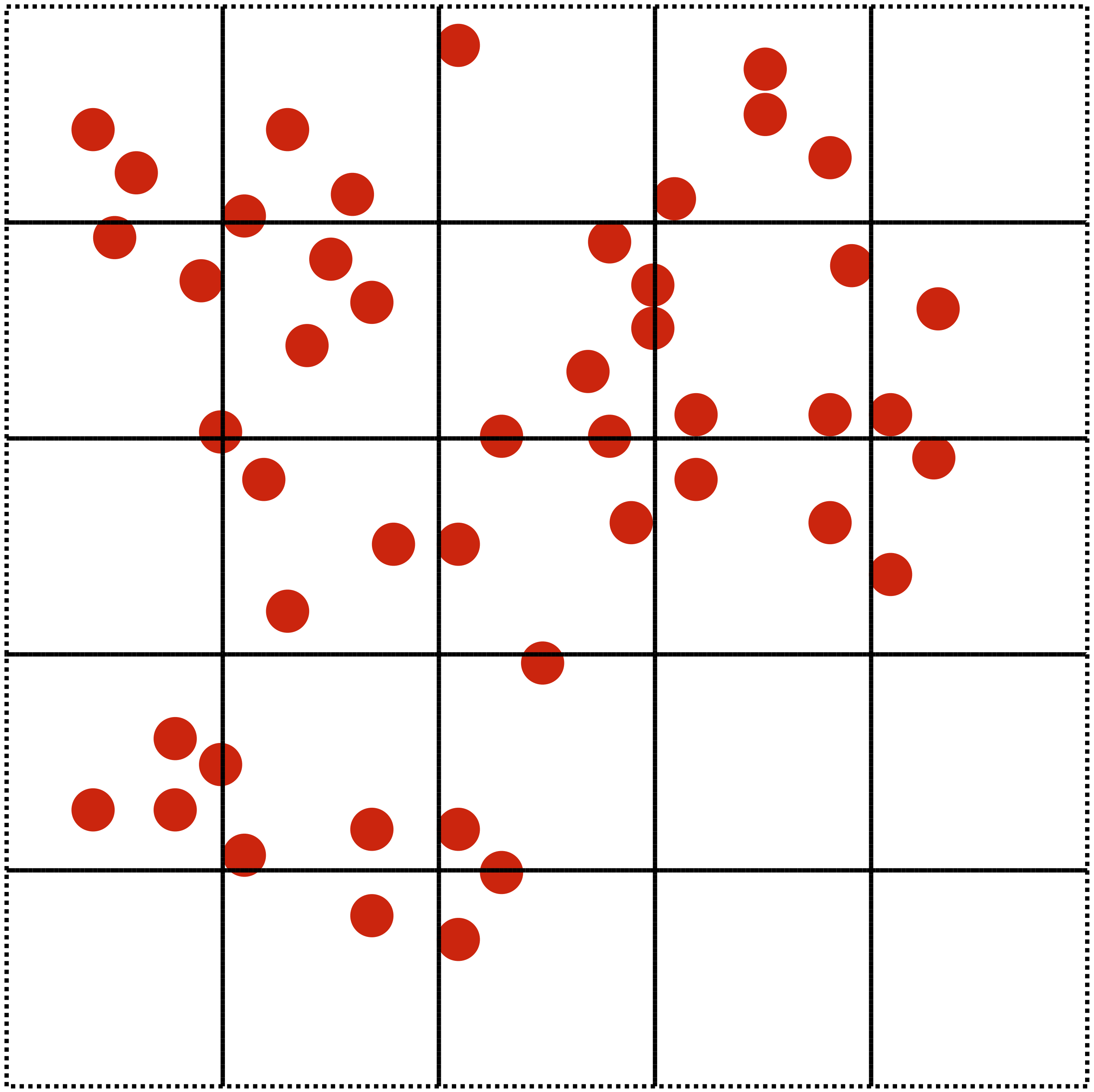
$$\nabla \times \mathbf{F}(\mathbf{r}) \equiv 0 \quad \Longrightarrow \quad \mathbf{F}(\mathbf{r}) = -\nabla\phi(\mathbf{r})$$

- Potential is given by Poisson's equation

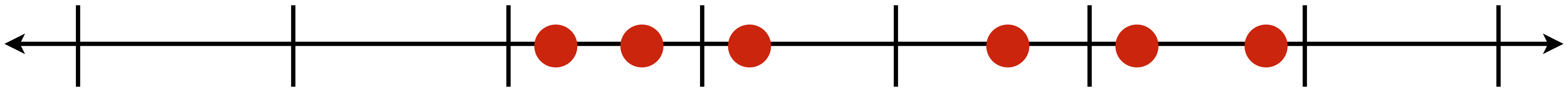
$$\nabla^2\phi(\mathbf{r}) = \rho(\mathbf{r})$$

- We know how to solve this! Just need a sensible **ρ**(r)

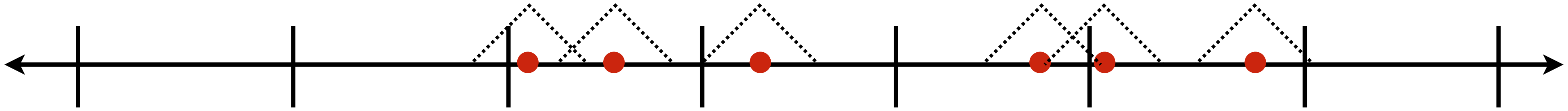
ρ: Create a mesh and assign particles to cells:



- **Nearest** grid point
  - Coarse ρ
- Charge-in-cell or particle-in-cell (**PIC**)
  - Assign “shape” or “cloud” to each particle
  - Smooths ρ
- **Boundary conditions?**
  - Periodic or multipole
- Last step: ρ → Φ → ***F***
  - Finite differencing + interpolation



Particle-in-cell:
Replace points by distribution function

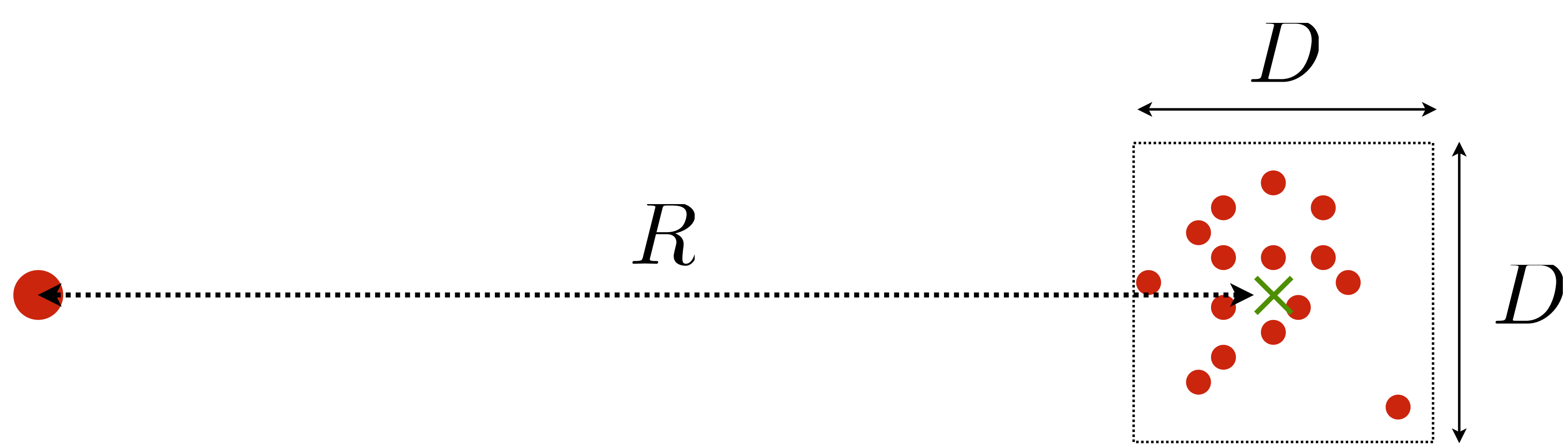


PM: Pros & cons

- Pro: Poisson is warm and fuzzy!
  - Many accurate and efficient solvers (*e.g.*, multigrid, FFT)
- Con: Limitations of meshes
  - How to choose no. of grid points?
  - Cannot resolve interactions on scales smaller than grid size
- Hybrid PP-PM (“P^3M”) methods possible

Efficiently computing forces: Tree codes

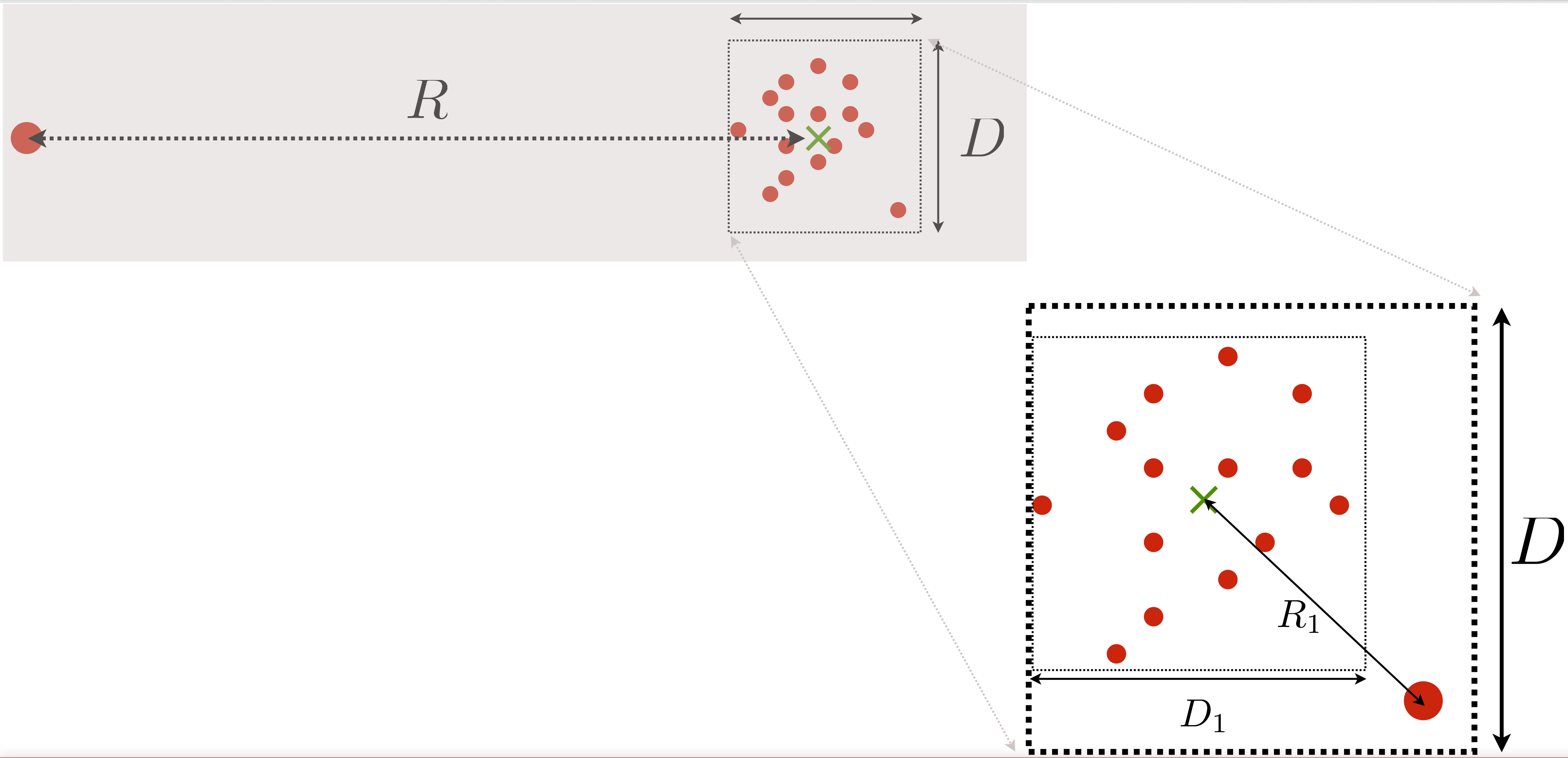
Approximating long-distance interactions



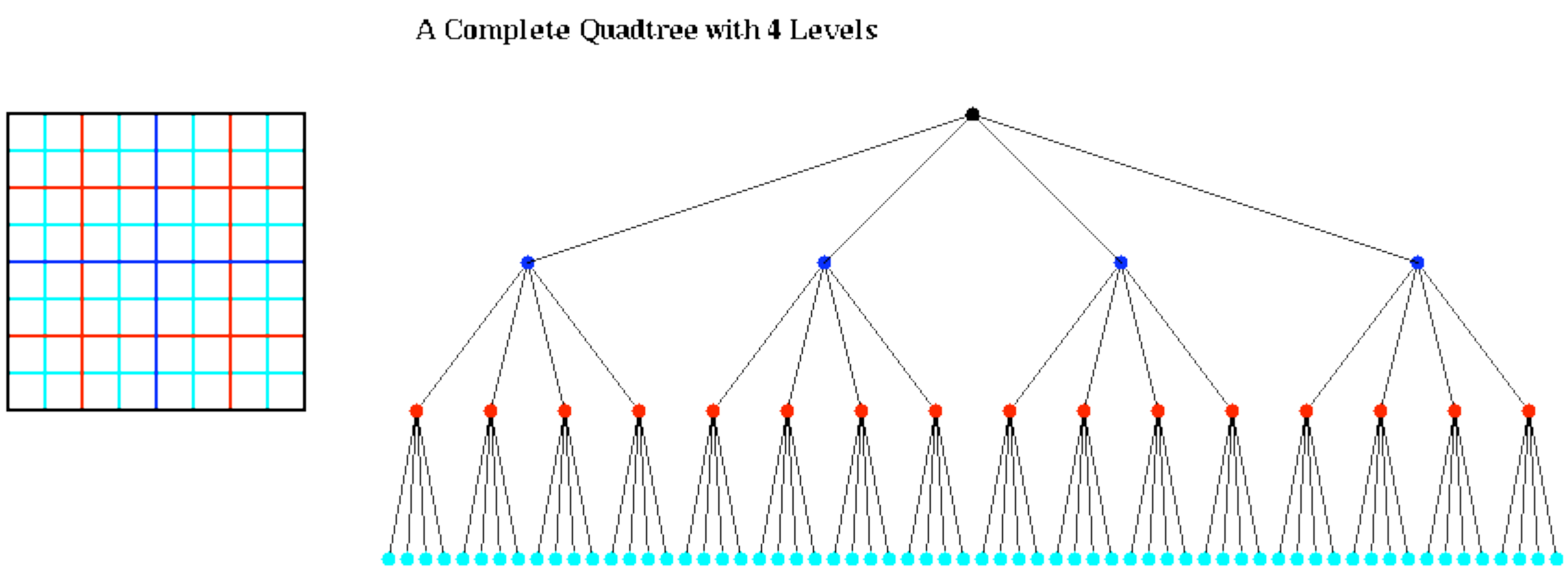
$$\frac{D}{R} < \theta$$

$\times$ = **center of mass**

Repeat recursively

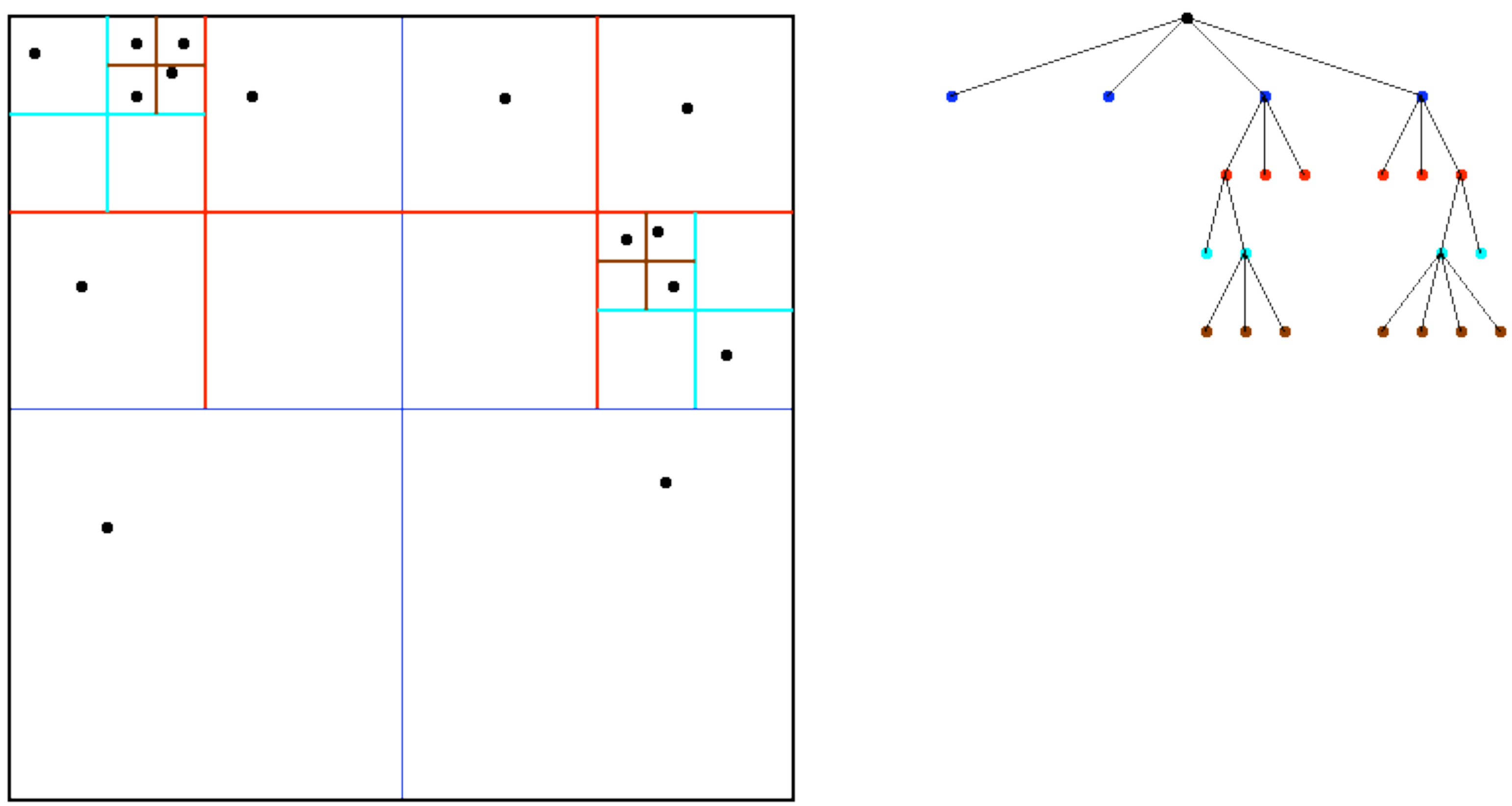


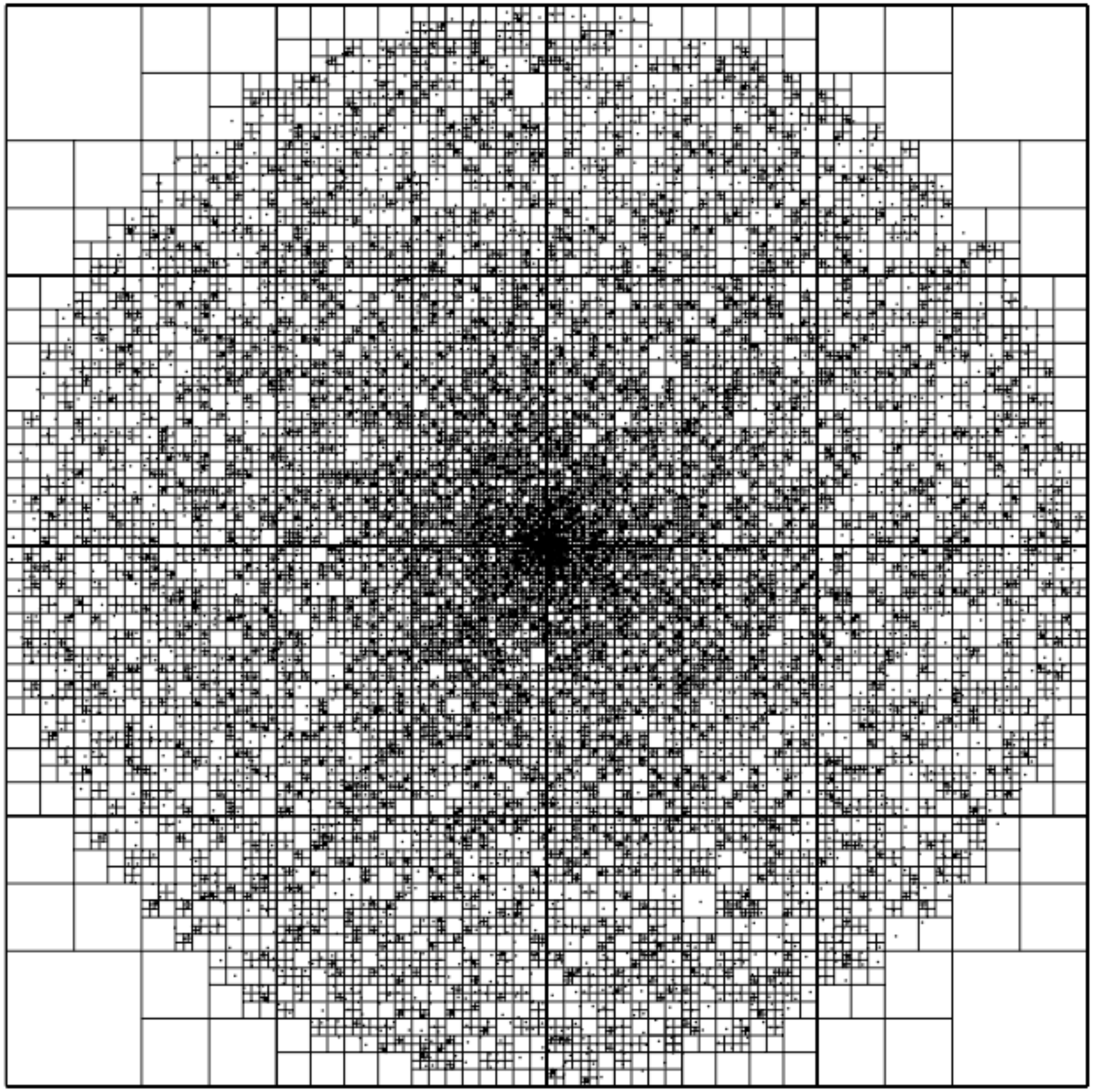
Idea: Organize particles in space in a tree



Adaptive quad-tree

Adaptive quadtree where no square contains more than 1 particle

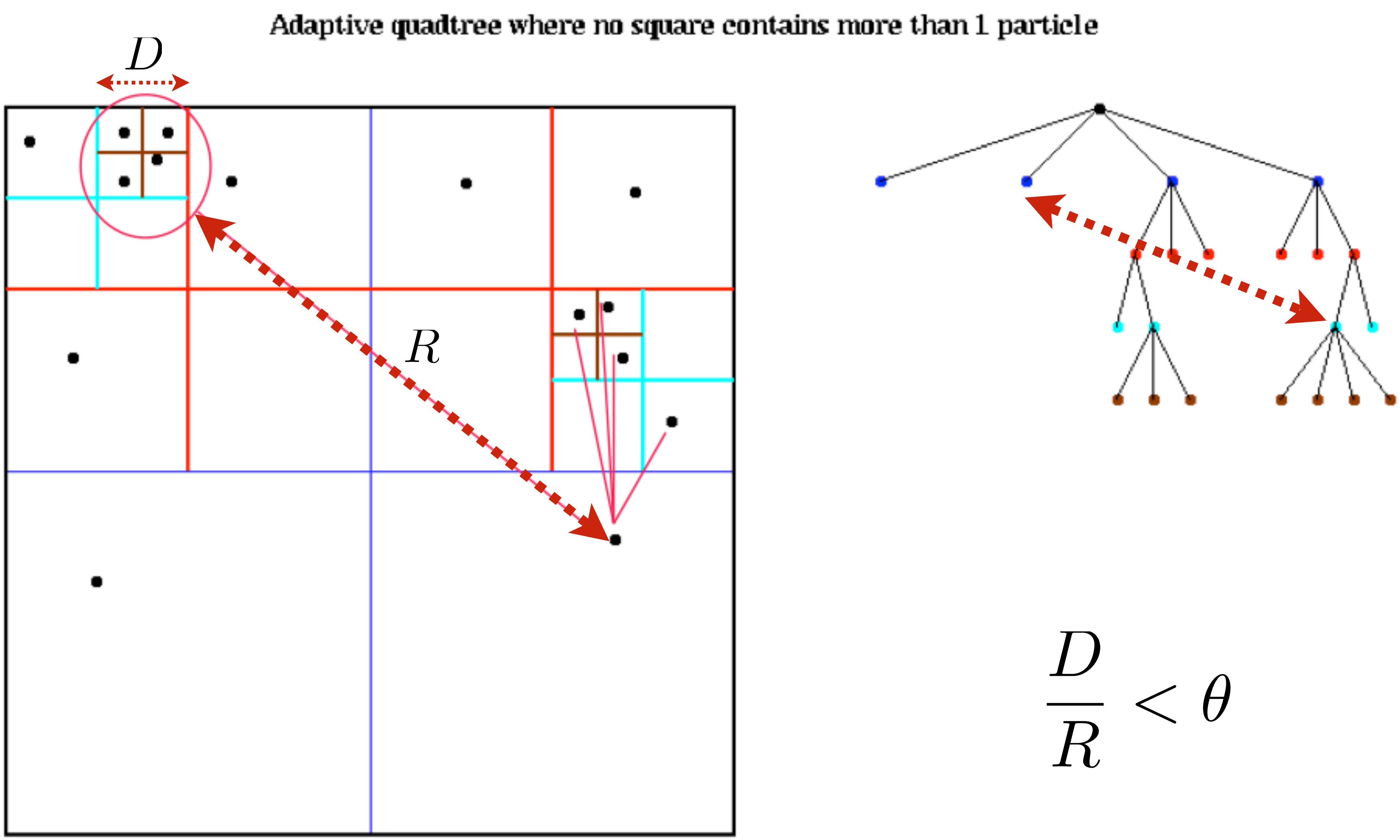




Source: M. Warren & J. Salmon, In Supercomputing 1993.

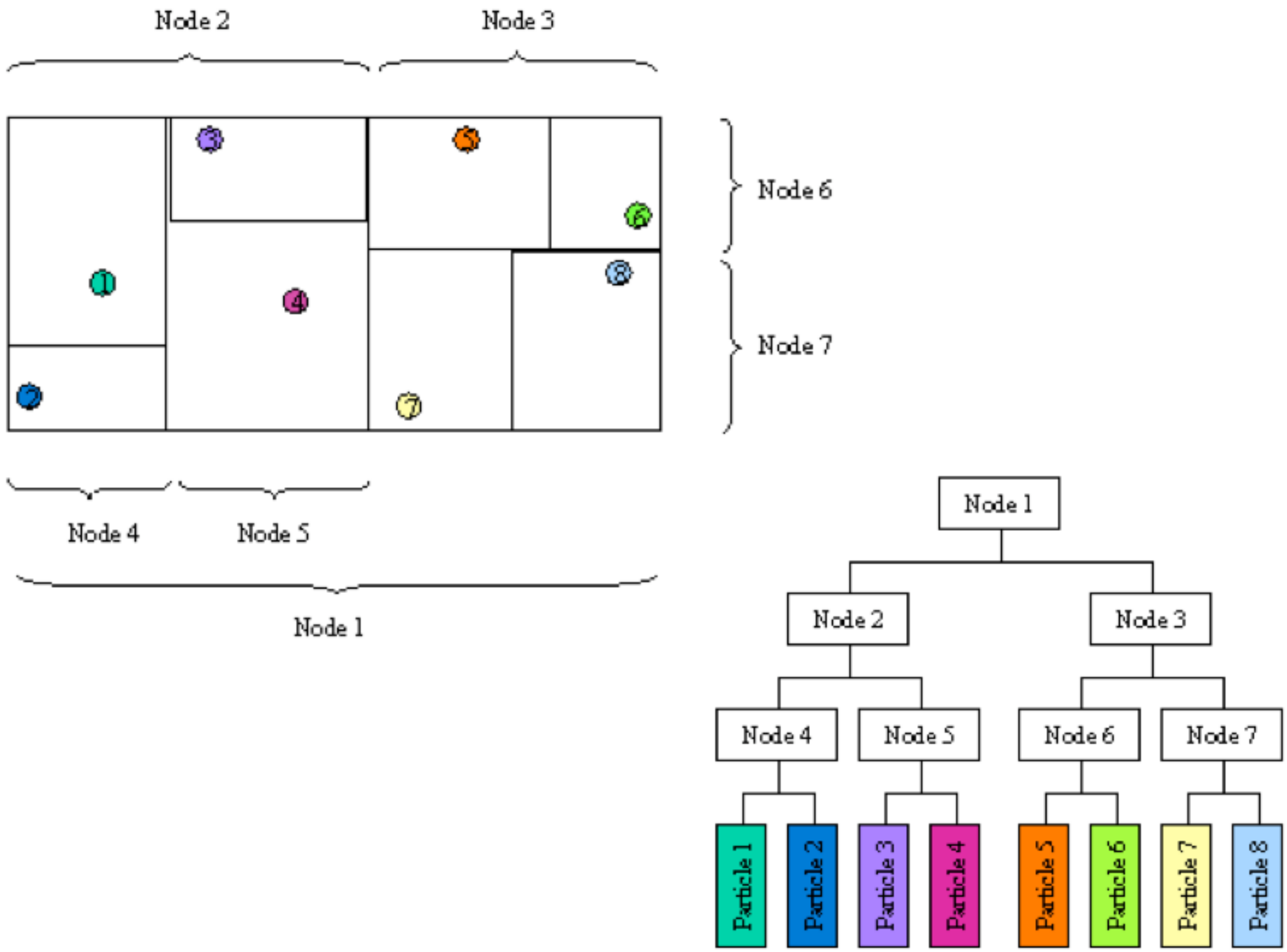
Barnes-Hut algorithm (1986)

- Algorithm:
  - **Build tree**
  - For each **node**, compute **center-of-mass and total mass**
  - For each **particle**, **traverse** tree to compute **force** on it



$$\frac{D}{R} < \theta$$

Other trees are possible, *e.g.*, kd-tree

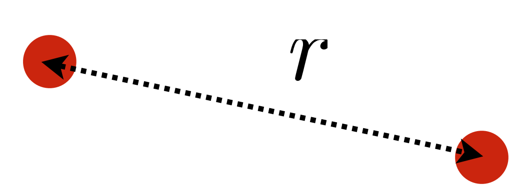


Fast multipole method of Greengard & Rokhlin (1987)

- Differences from Barnes-Hut
  - Computes potential, not force
  - Uses more than center-of- and total-mass ⇒ more accurate & expensive
  - Accesses fixed set of boxes at every level, independent of “D / R”
- Increasing accuracy
  - BH: Fixed info / box, more boxes
  - FMM: Fixed no. of boxes; more info / box

FMM computes compact expression for potential

$$
\begin{aligned}
|\mathbf{F}(\mathbf{r})| &= \frac{1}{r^2} \\
&\Downarrow \\
\mathbf{F}(\mathbf{r}) &= -\nabla\phi(\mathbf{r})
\end{aligned}
$$



Potential in 3-D

3-D:

$$\phi(\mathbf{r}) = -\frac{1}{|\mathbf{r}|} = -\frac{1}{\sqrt{x^2+y^2+z^2}}$$

$$\mathbf{F}(\mathbf{r}) = -\left(\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}, \frac{\partial\phi}{\partial z}\right) = -\left(\frac{x}{r^3}, \frac{y}{r^3}, \frac{z}{r^3}\right)$$

Potential in 2-D

2-D:

$$\begin{aligned} \phi(\mathbf{r}) &= \ln|\mathbf{r}| = \ln\sqrt{x^2+y^2} \\ \mathbf{F}(\mathbf{r}) &= -\left(\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}\right) = -\left(\frac{x}{r^2}, \frac{y}{r^2}\right) \end{aligned}$$

For *n* points in the plane:

$$\phi(x,y) = \sum_{k=1}^{n} m_k \ln\sqrt{(x-x_k)^2+(y-y_k)^2}$$

“Complex” representation

2-D:

$$\phi(x, y) = \sum_{k=1}^{n} m_k \ln \sqrt{(x - x_k)^2 + (y - y_k)^2}$$

Complex plane:

$$\begin{aligned} z &\equiv x + iy \\ \phi(z) &= \sum_{k=1}^{n} m_k \ln |z - z_k| \\ &= \mathrm{Re}\left\{ \sum_{k=1}^{n} m_k \ln(z - z_k) \right\} \end{aligned}$$

2-D multipole expansion

$$
\begin{aligned}
\sum_{k=1}^{n} m_k \ln(z - z_k) &= M \ln z + \sum_k m_k \ln\left(1 - \frac{z_k}{z}\right) \\
&= M \ln z + \sum_k m_k \sum_{d=1}^{\infty} \frac{1}{d} \left(\frac{z_k}{z}\right)^d \\
&= M \ln z + \sum_{d=1}^{\infty} \underbrace{\left(\sum_{k=1}^{n} m_k z_k^d\right)}_{\equiv \alpha_d} \frac{1}{z^d} \\
&= M \ln z + \sum_{d=1}^{\infty} \frac{\alpha_d}{z^d}
\end{aligned}
$$

Can approx. by truncation

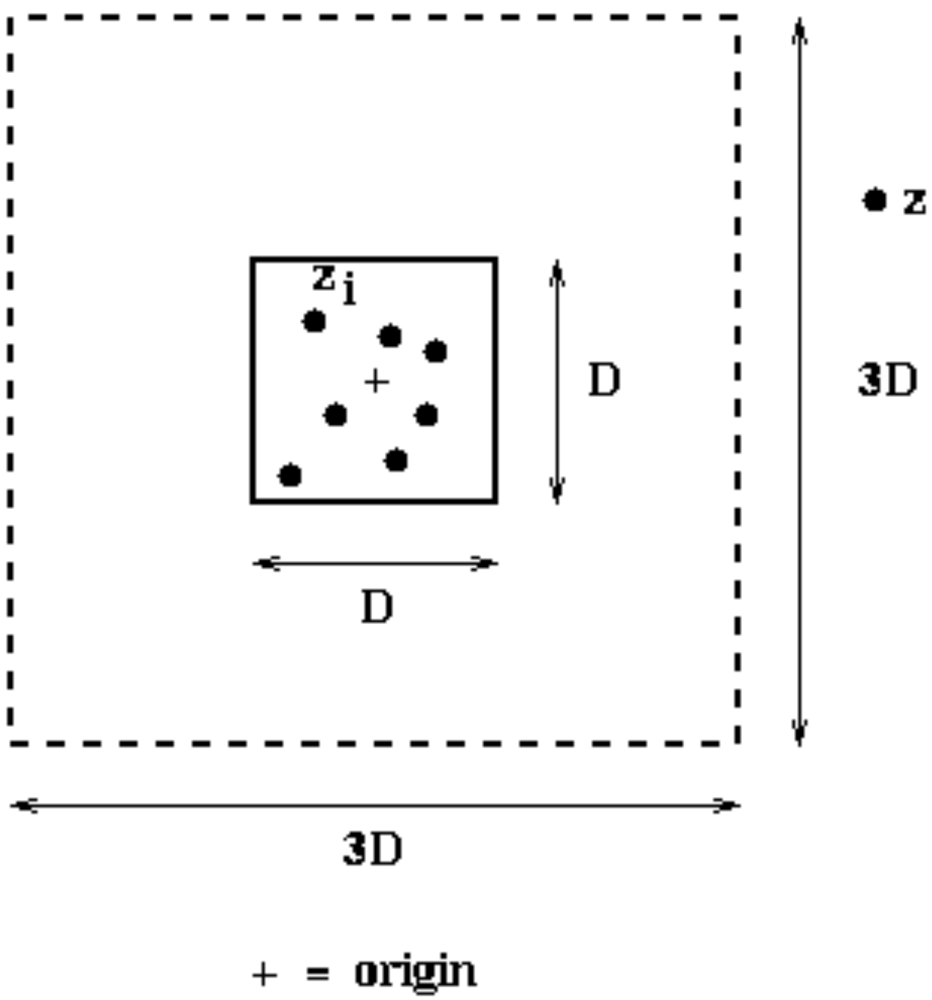
2-D multipole expansion

$$
\begin{aligned}
\alpha_d &\equiv \sum_{k=1}^{n} m_k z_k^d \\
\sum_{k=1}^{n} m_k \ln(z - z_k) &= M \ln z + \sum_{d=1}^{\infty} \frac{\alpha_d}{z^d} \\
&\approx M \ln z + \sum_{d=1}^{p} \frac{\alpha_d}{z^d} + \mathrm{Error}(p)
\end{aligned}
$$

$$
\mathrm{Error}(p) \sim \left(\frac{\max |z_k|}{|z|} \right)^{p+1}
$$

$$\mathrm{Error}(p) \quad \sim \quad \left(\frac{\max |z_k|}{|z|} \right)^{p+1}$$

Error outside larger box is $O(c^{p+1})$



$$c \sim \frac{\frac{D}{\sqrt{2}}}{\frac{3}{2}D} \approx .47$$

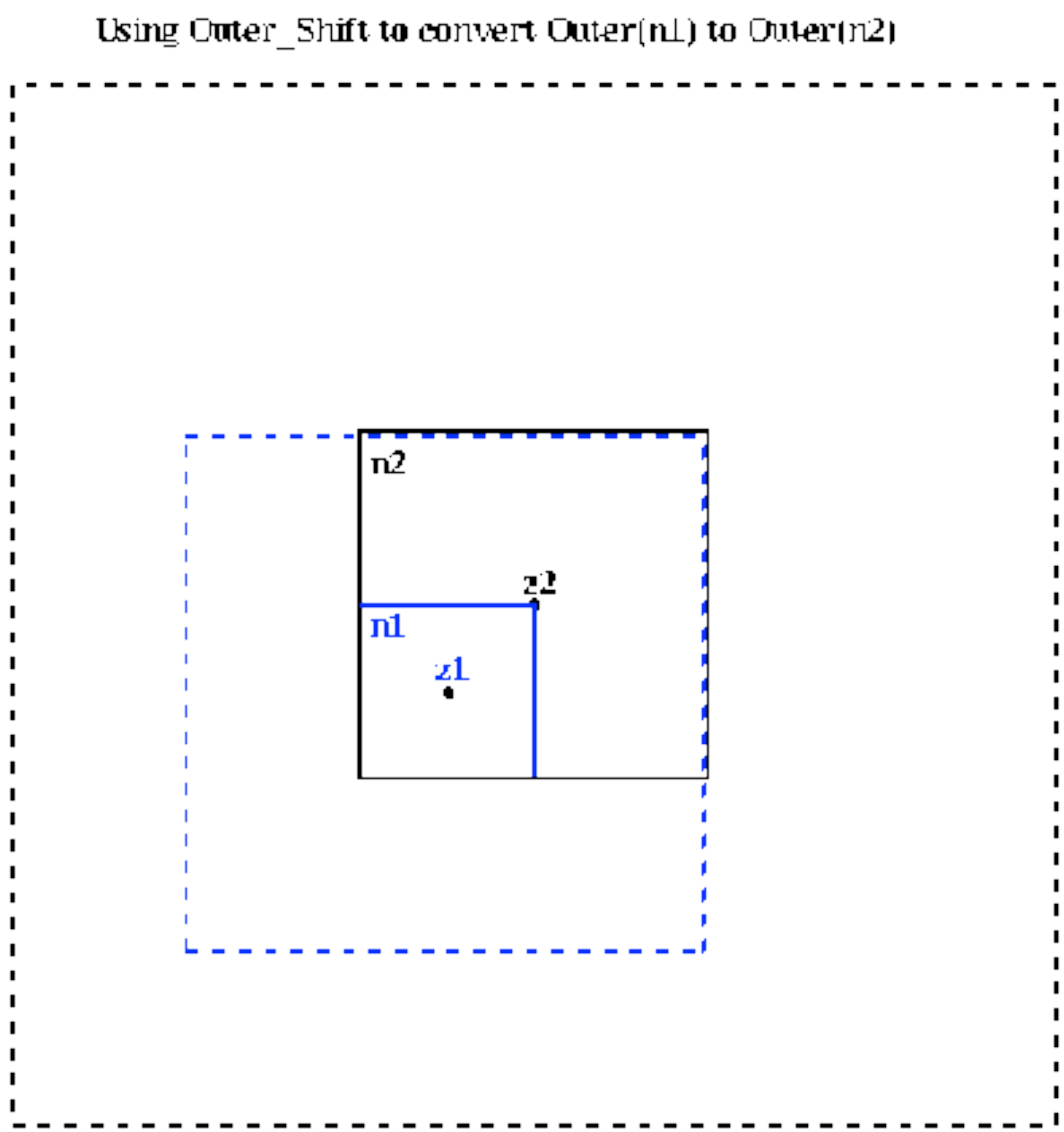
Outer expansion

$$\alpha_d \equiv \sum_{k=1}^{n} m_k z_k^d$$

$$\sum_{k=1}^{n} m_k \ln(z - z_k) \approx M \ln z + \sum_{d=1}^{p} \frac{\alpha_d}{z^d}$$

- **Outer(N)** = {M, α_1, α_2, …, α_p, z_N}
  - Use to evaluate $\phi(z)$ **outside** node N due to those inside N
  - Centered at z_N
  - Evaluation costs is O(p)
  - Cost linear with no. of bits of precision

$$\sum_{k=1}^{n} m_k \ln(z - z_k) \quad \approx \quad M \ln z + \sum_{d=1}^{p} \frac{\alpha_d}{z^d}$$



$$\phi_1(z) = M_1 \ln(z - z_1) + \sum_{d=1}^{p} \frac{\alpha_d^{(1)}}{(z - z_1)^d}$$

$$\phi_2(z) = M_2 \ln(z - z_2) + \sum_{d=1}^{p} \frac{\alpha_d^{(2)}}{(z - z_2)^d}$$

For z outside dashed black box:

$$\phi_1(z) \sim \phi_2(z)$$

$$\Longrightarrow \begin{pmatrix} \alpha_1^{(2)} \\ \vdots \\ \alpha_d^{(2)} \end{pmatrix} \approx A(z_1) \cdot \begin{pmatrix} \alpha_1^{(1)} \\ \vdots \\ \alpha_d^{(1)} \end{pmatrix}$$

- **Outer(N2)** = **Outer_shift(Outer(N1), z_2)**

Inner expansion

- **Outer(N)** = {M, α_1, α_2, …, α_p, z_N}

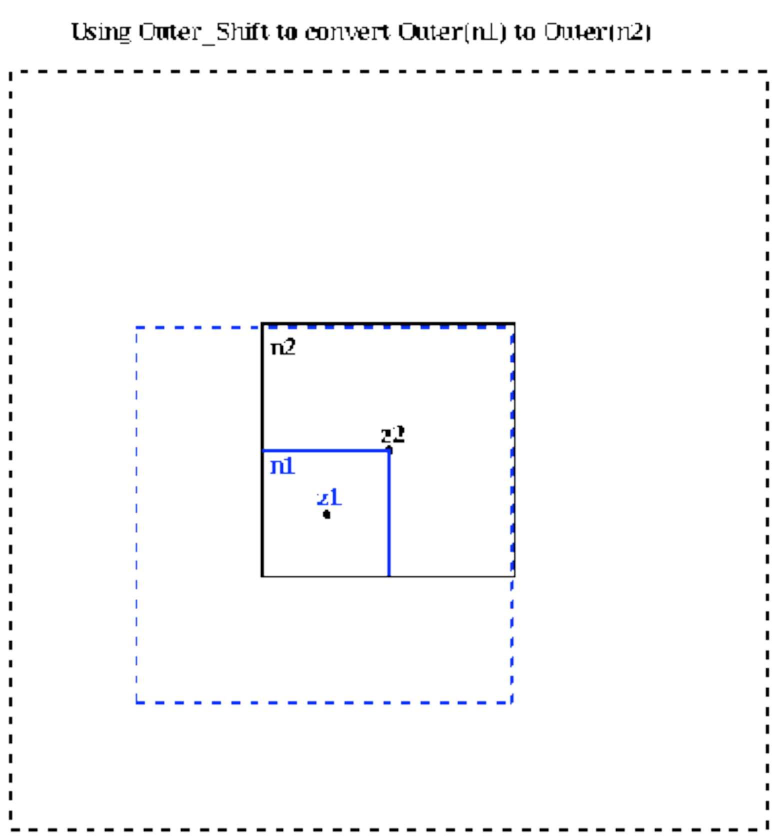
$$\sum_{k=1}^{n} m_k \ln(z - z_k) \quad \approx \quad M \ln z + \sum_{d=1}^{p} \frac{\alpha_d}{z^d}$$

- Similarly, **Inner(N)** evaluates potential inside N from particles outside.

$$\sum_{d=1}^{p} \beta_d (z - z_N)^d$$

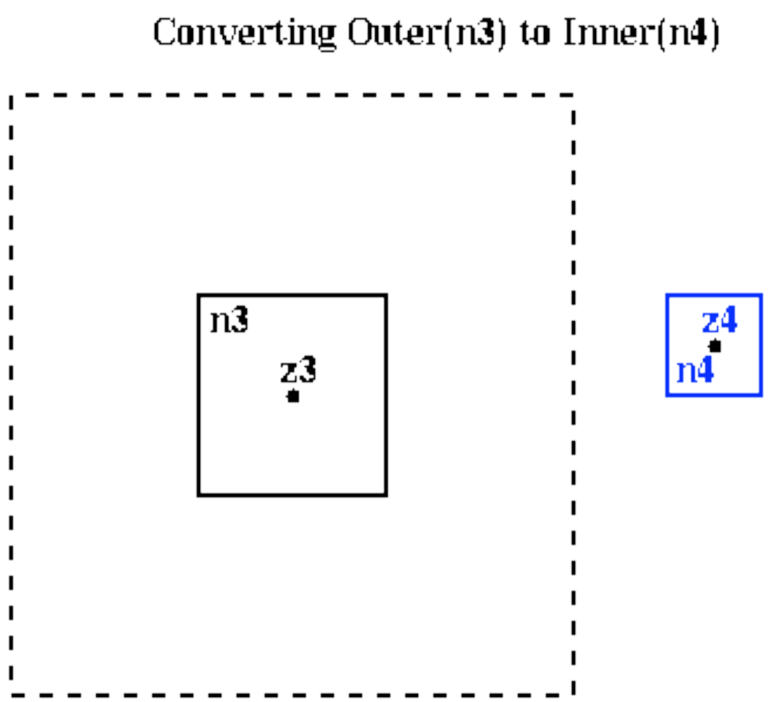
Inner expansion

Inner(N1) = **Inner_shift(Inner(N2), z_1)**



Need Inner(N4) =

Convert(Outer(N3))



FMM algorithm

- Build tree
- Bottom-up traversal to compute Outer(N)
- Top-down traversal to compute Inner(N)
- For each leaf N, add contributions of nearest particles directly into Inner(N)

FMM algorithm

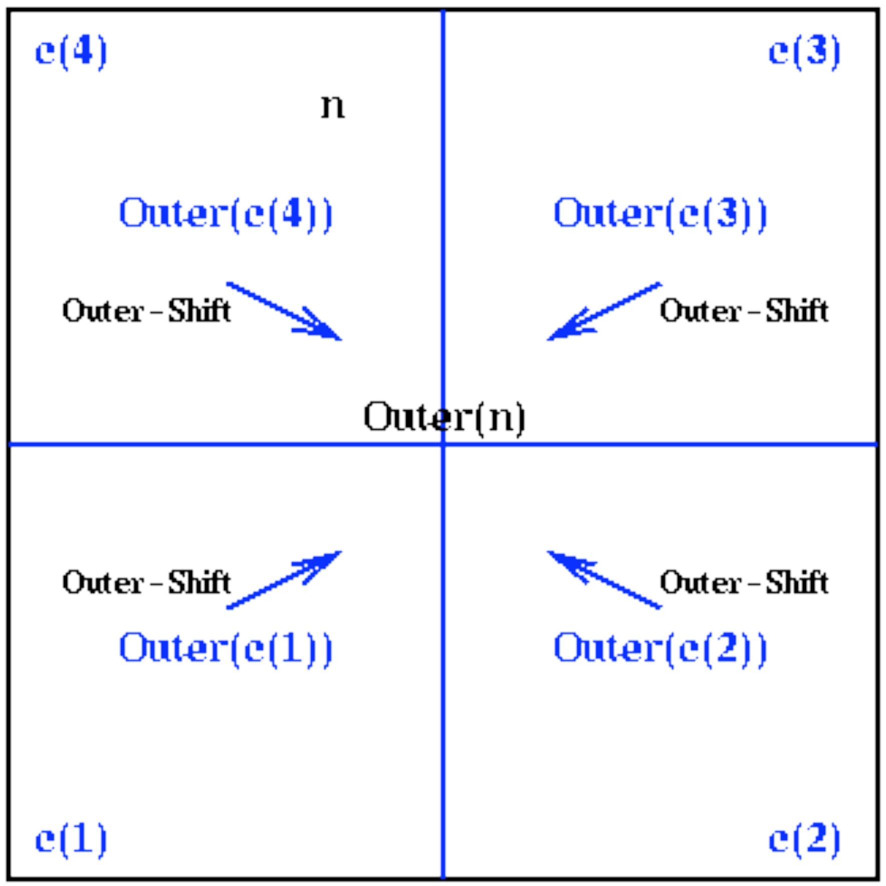
- **Build tree**
- Bottom-up traversal to compute Outer(N)
- Top-down traversal to compute Inner(N)
- For each leaf N, add contributions of nearest particles directly into Inner(N)

FMM algorithm

- Build tree
- **Bottom-up traversal to compute Outer(N)**
- Top-down traversal to compute Inner(N)
- For each leaf N, add contributions of nearest particles directly into Inner(N)

Building Outer(N)

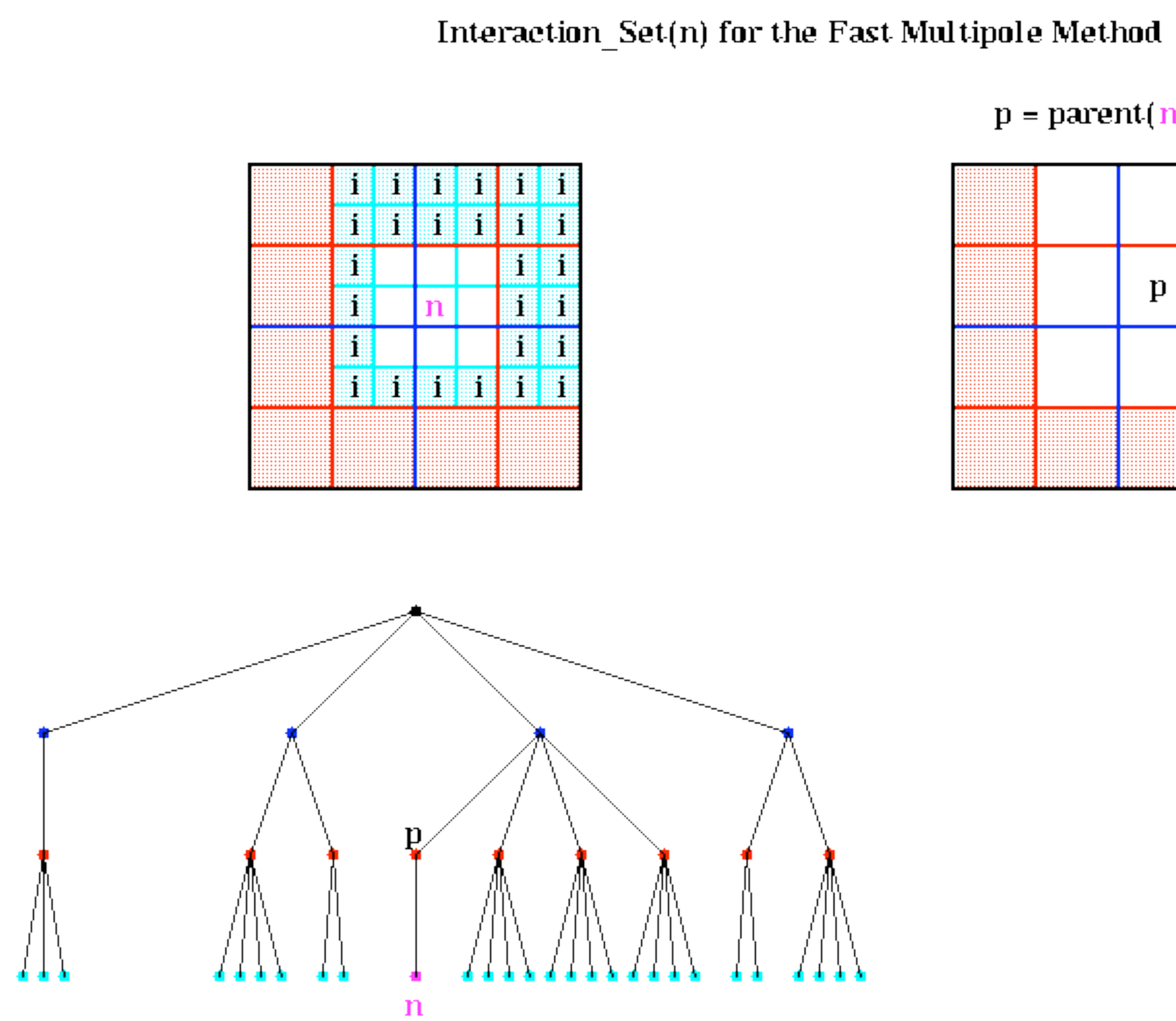
Inner Loop of Build_Outer



FMM algorithm

- Build tree
- Bottom-up traversal to compute Outer(N)
- **Top-down traversal to compute Inner(N)**
- For each leaf N, add contributions of nearest particles directly into Inner(N)

Building Inner(N)



FMM algorithm

- Build tree
- Bottom-up traversal to compute Outer(N)
- Top-down traversal to compute Inner(N)
- **For each leaf N, add contributions of nearest particles directly into Inner(N)**

Administrivia

Upcoming schedule changes

- Some adjustment of topics (TBD)
- **Tu 4/1 — Esteemed colleague**
  - **R. Riegel on "Dual tree algorithms in statistics"**
- Th 4/3 — Attend talk by Dr. Douglass Post from DoD HPC Modernization Program, 9:30–10:30am, room MiRC 102A&B
- No HW 2 (optional assignment possible)
- Project checkpoint (3 page max): Tu 4/8

“In conclusion…”

Ideas apply broadly

- Physical sciences, *e.g.*,
  - Plasmas
  - Molecular dynamics
  - Electron-beam lithography device simulation
  - Fluid dynamics
- “Generalized” n-body problems: Talk to your classmate, Ryan Riegel

Backup slides

2-D multipole expansion

$$\phi(z) = \mathrm{Re}\left\{\sum_{k=1}^{n} m_k \ln(z - z_k)\right\}$$

$$\Downarrow$$

$$\sum_{k=1}^{n} m_k \ln(z - z_k) = M \ln z + \sum_{k} m_k \ln\left(1 - \frac{z_k}{z}\right)$$

$$= M \ln z + \underline{\sum_{d=1}^{\infty} \frac{\alpha_d}{z^d}}$$

Approx. by truncating sum