

Computational Astrophysics

Lecture 4: Particles and Gravity

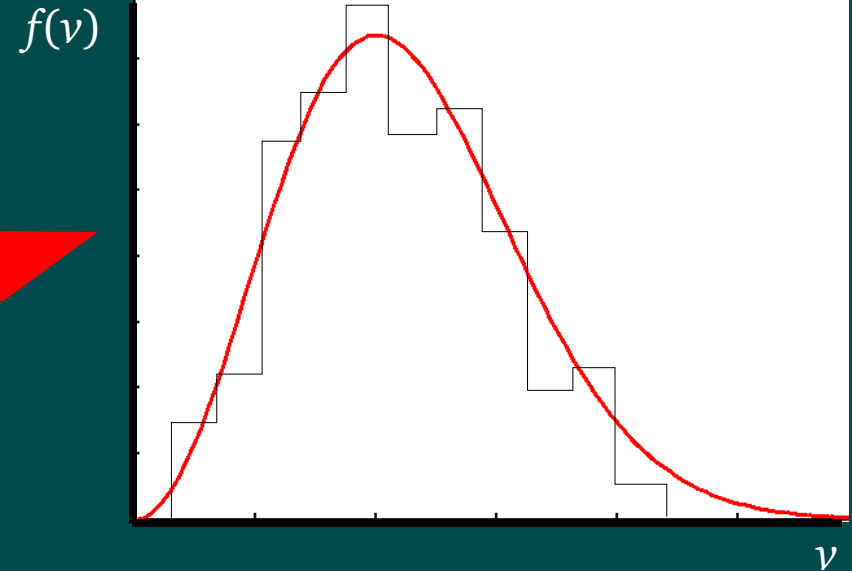
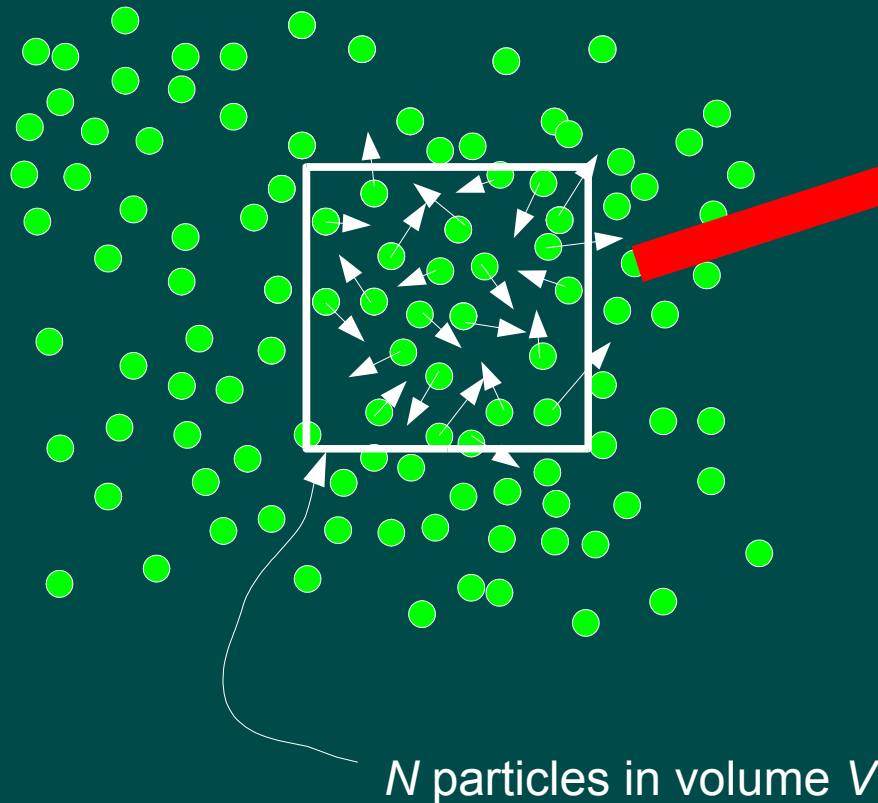
Paul Ricker

University of Illinois at Urbana-Champaign
National Center for Supercomputing Applications
Urbana, Illinois USA



Particle simulation

Monte Carlo sampling of particle distribution function (gas, dust, dark matter)



$$n(\mathbf{x}) = \int f(\mathbf{x}, v) dv \approx \frac{N}{V}$$

$$\frac{|f_N - f_{\text{true}}|}{|f_{\text{true}}|} \propto N^{-1/2}$$

Basic requirements:

- As $N \rightarrow \infty$, error (“shot noise”) in approximate distribution function f_N goes to 0
- As $N \rightarrow \infty$, equation describing evolution of f_N becomes the Boltzmann equation

Equations of motion

- Consider particle interactions due to gravitation
- Consider a system of N particles with masses m_i , positions \mathbf{x}_i , and velocities \mathbf{v}_i , moving under the influence of an external potential ϕ_{ext} and their mutual gravitational attraction:

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i$$

$$\frac{d\mathbf{v}_i}{dt} = -\nabla \phi_{\text{ext}} - G \sum_{\substack{j=1 \\ j \neq i}}^N m_j \frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|^3}$$

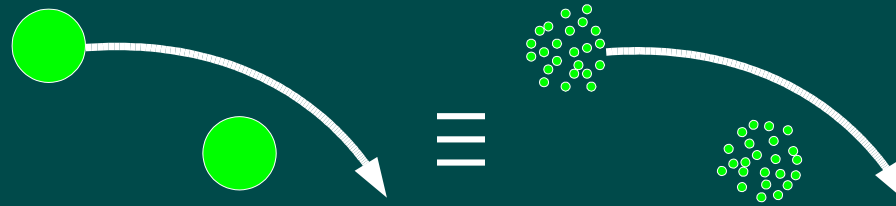
- Start with known \mathbf{x}_i and \mathbf{v}_i , then march forward in time

Particle collisions

- Collisionless particles: solving the Vlasov-Poisson equation:

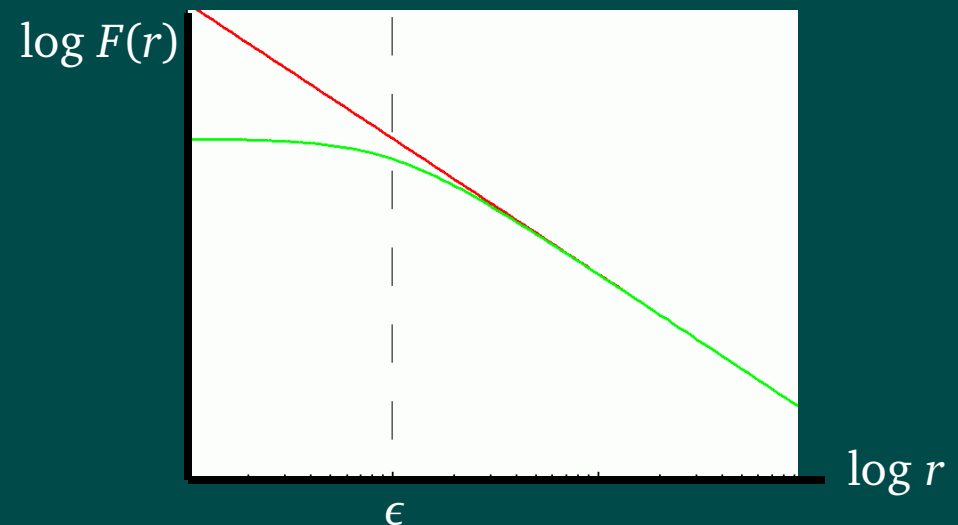
$$\frac{\partial f(\mathbf{x}, \mathbf{v})}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{v}) - Gm \left[\int d^3 x' \frac{\mathbf{x} - \mathbf{x}'}{|\mathbf{x} - \mathbf{x}'|^3} \int d^3 v' f(\mathbf{x}', \mathbf{v}') \right] \cdot \nabla_{\mathbf{v}} f(\mathbf{x}, \mathbf{v}) = 0$$

- Want to avoid two-body relaxation



- Softened force law:

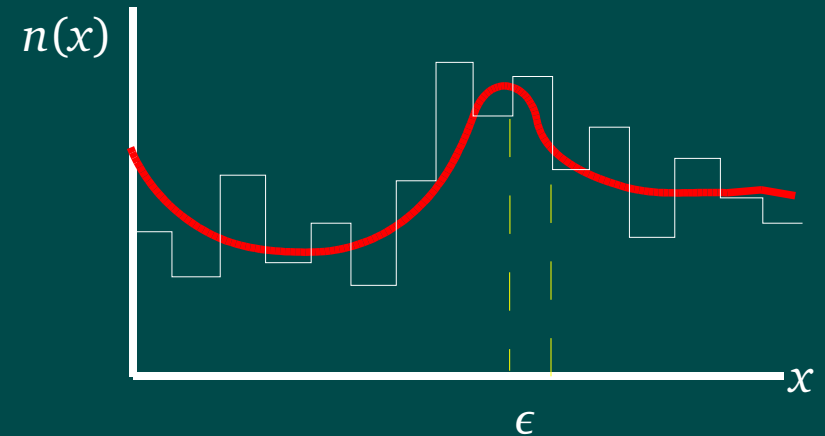
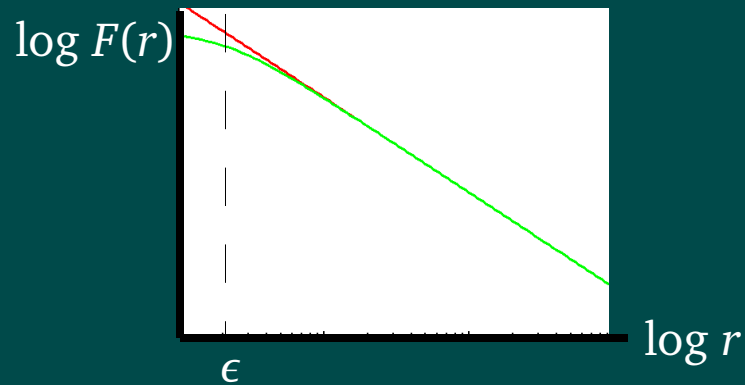
$$F(r) = -\frac{Gm^2}{r^2 + \epsilon^2}$$



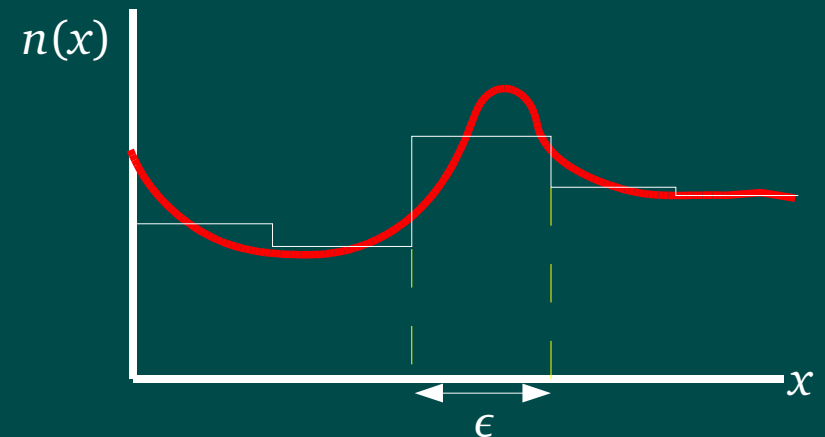
Particle collisions

Maximizing force resolution and minimizing shot noise require a tradeoff:

If we fix N and reduce ϵ , we get better force resolution, but we increase unphysical collision effects, and we do not improve the distribution sampling accuracy.



If we fix ϵ and increase N , we minimize collision effects and shot noise, but we do not improve force resolution, and features smaller than ϵ cannot be resolved.



So typically we want to increase N and decrease ϵ in concert: $N\epsilon^3 \sim \text{constant}$.

Progress in N -body simulation

E. Holmberg (1941) – using light bulbs and photocells

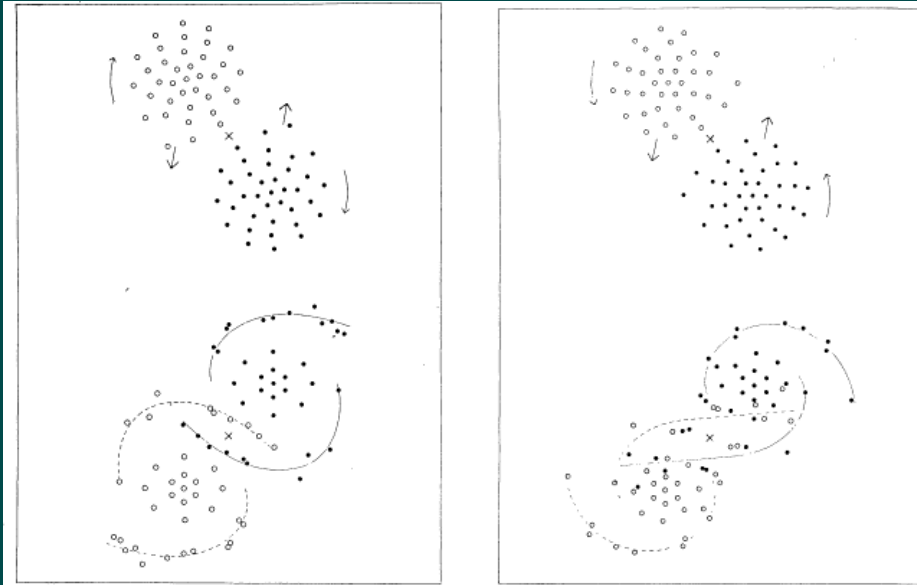
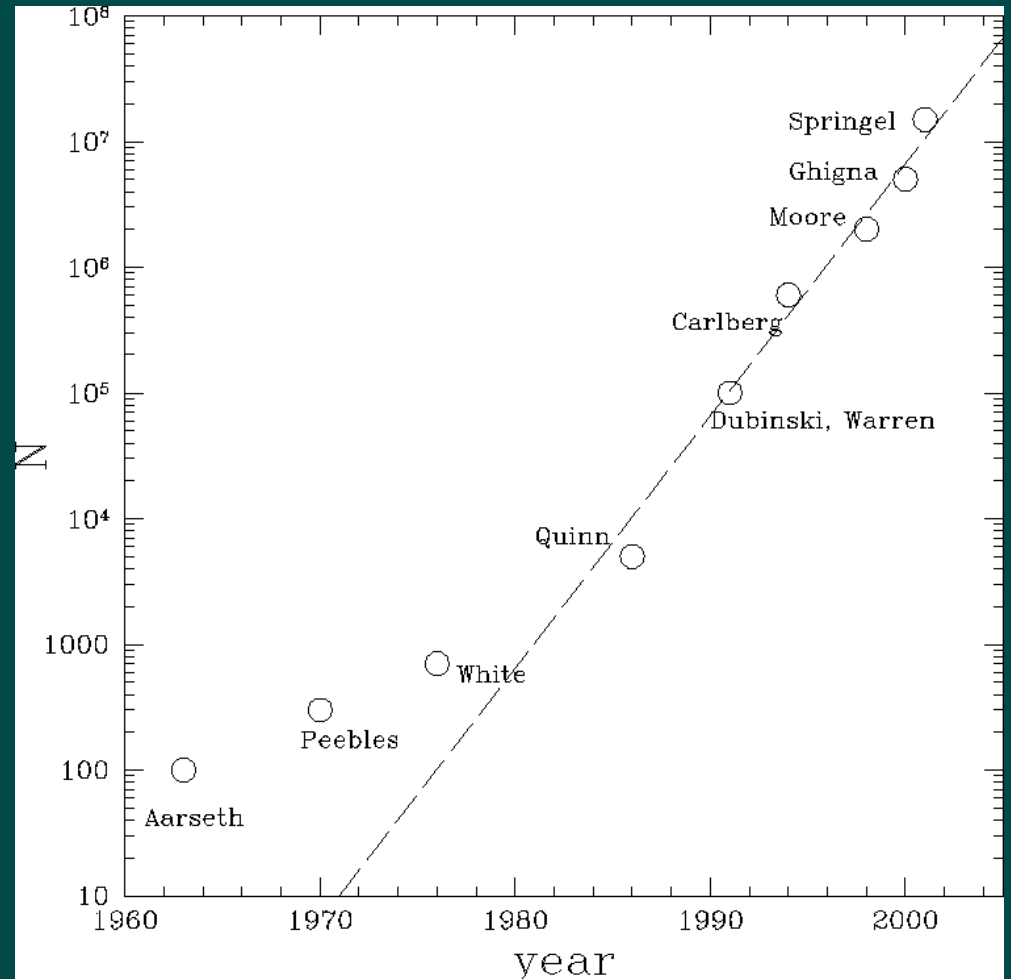


FIG. 4a.—Tidal deformations corresponding to parabolic motions, clockwise rotations, and a distance of closest approach equal to the diameters of the nebulae. The spiral arms point in the direction of the rotation.

FIG. 4b.—Same as above, with the exception of counterclockwise rotations. The spiral arms point in the direction opposite to the rotation.



B. Moore (2000)

Pieces of a particle simulation algorithm

Given particle positions \mathbf{x}_i^n and velocities \mathbf{v}_i^n at time t_n , compute values at time t_{n+1} :

1. Compute acceleration of each particle at t_n : \mathbf{a}_i^n
2. Using \mathbf{a}_i^n and possibly acceleration values from previous steps, advance \mathbf{v}_i^n :

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \Delta t \mathbf{a}_{\text{eff},i}(\mathbf{a}_i^n, \dots)$$

3. Using \mathbf{v}_i^n and possibly velocity values from previous steps, advance \mathbf{x}_i^n :

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \Delta t \mathbf{v}_{\text{eff},i}(\mathbf{v}_i^n, \dots)$$

Positions and velocities may be evaluated at different times (e.g., t_n and $t_{n+1/2}$).

Basic components:

- Field computation
 - Direct N -body (particle-particle)
 - Particle-mesh
 - Particle-particle-particle mesh (P³M)
 - Treecodes
- Time integrator

Time integration methods

Basically an ODE integration problem, but some special requirements lead us to prefer particular solvers:

- We require at least $O(\Delta t^2)$ accuracy.
- If N is large, we would like to avoid implicit methods.
- We would also like to avoid methods that involve multiple evaluations of the acceleration at each step (since these are expensive).
- If we must integrate for a large number of timesteps, our method should explicitly conserve energy, or else the accumulation of roundoff error will make the solution meaningless.

Leapfrog (Størmer-Verlet) method

Positions and velocities are staggered in time (hence the name):

$$\mathbf{v}_i^{n+1/2} = \mathbf{v}_i^{n-1/2} + \frac{1}{m_i} \mathbf{F}(\mathbf{x}_i^n) \Delta t$$

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \mathbf{v}_i^{n+1/2} \Delta t$$

This method has $O(\Delta t^2)$ truncation error: eliminate \mathbf{v} (1D, drop particle index i):

$$\frac{x^{n+1} - 2x^n + x^{n-1}}{\Delta t^2} = \frac{F(x^n)}{m}$$

The true solution X satisfies

$$\frac{d^2 X}{dt^2} = \frac{F}{m}$$

The truncation error δ is obtained when we substitute X into the update method and use Taylor expansions about X^n to substitute for X^{n+1} and X^{n-1} :

$$\frac{X^{n+1} - 2X^n + X^{n-1}}{\Delta t^2} = \frac{F(X^n)}{m} - \delta^n$$

$$\frac{d^2 X^n}{dt^2} + \frac{\Delta t^2}{12} \frac{d^4 X^n}{dt^4} + \dots = \frac{F(X^n)}{m} - \delta^n \quad \Rightarrow \quad \delta^n \propto \Delta t^2$$

Particle-particle (PP) or direct N -body

Simplest possible N -body technique.

1. For each particle i , compute $\mathbf{a}_i = -G \sum_{\substack{j=1 \\ j \neq i}}^N m_j \frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|^3}$

2. For each particle, integrate $\frac{d(\mathbf{x}_i, \mathbf{v}_i)}{dt} = (\mathbf{v}_i, \mathbf{a}_i)$
from t_n to $t_n + \Delta t$.

Advantages:

- Force law is exact \rightarrow good if collisions important
- No restrictions on geometry or boundary conditions

Disadvantage:

- Force computation scales as N^2

PP is usually combined with individual particle timesteps and special treatment of close encounters and binary stars.



GRAPE (GRAvity PipE) special-purpose force-summation computer
J. Makino et al. – University of Tokyo

Particle-mesh method

- Exploit fast mesh-based Poisson solvers
- First introduced for plasma simulation in the early 1960s
- Procedure:
 1. Assign particle masses to mesh using a mass assignment operator $\rightarrow \rho_{ijk}$.
 2. Solve $\nabla^2 \phi = 4\pi G \rho$ on mesh.
 3. Finite-difference ϕ to get forces on mesh.
 4. Interpolate forces to the particle positions using a force interpolation operator.
 5. Advance the positions and velocities of the particles in time.
 6. Repeat.

Particle-mesh transfer operators

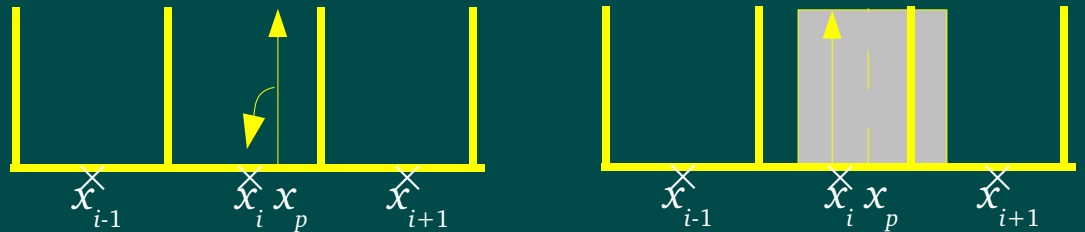
Mass assignment operator:

$$\rho_{ijk} = \frac{m}{\Delta x \Delta y \Delta z} \sum_{p=1}^{N_p} W(x_i - x_p) W(y_j - y_p) W(z_k - z_p)$$

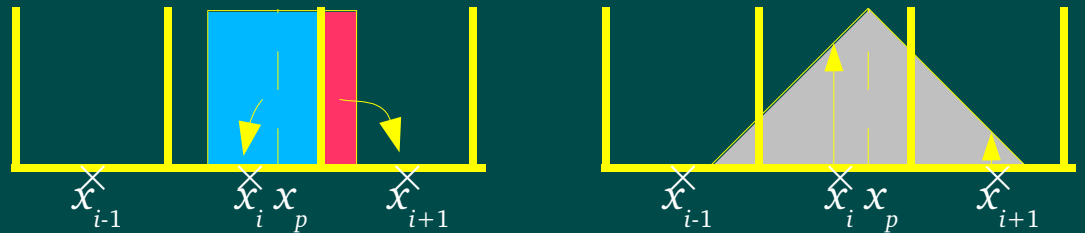
Force interpolation operator:

$$\mathbf{F}(x_p) = -m \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \sum_{k=1}^{N_g} (\nabla \phi)_{ijk} W(x_p - x_i) W(y_p - y_j) W(z_p - z_k)$$

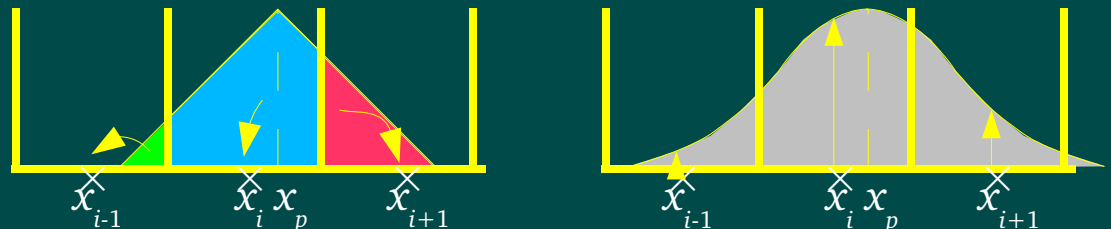
Nearest grid point (NGP)



Cloud in cell (CIC)



Triangle-shaped cloud (TSC)



Force smoothing in particle-mesh

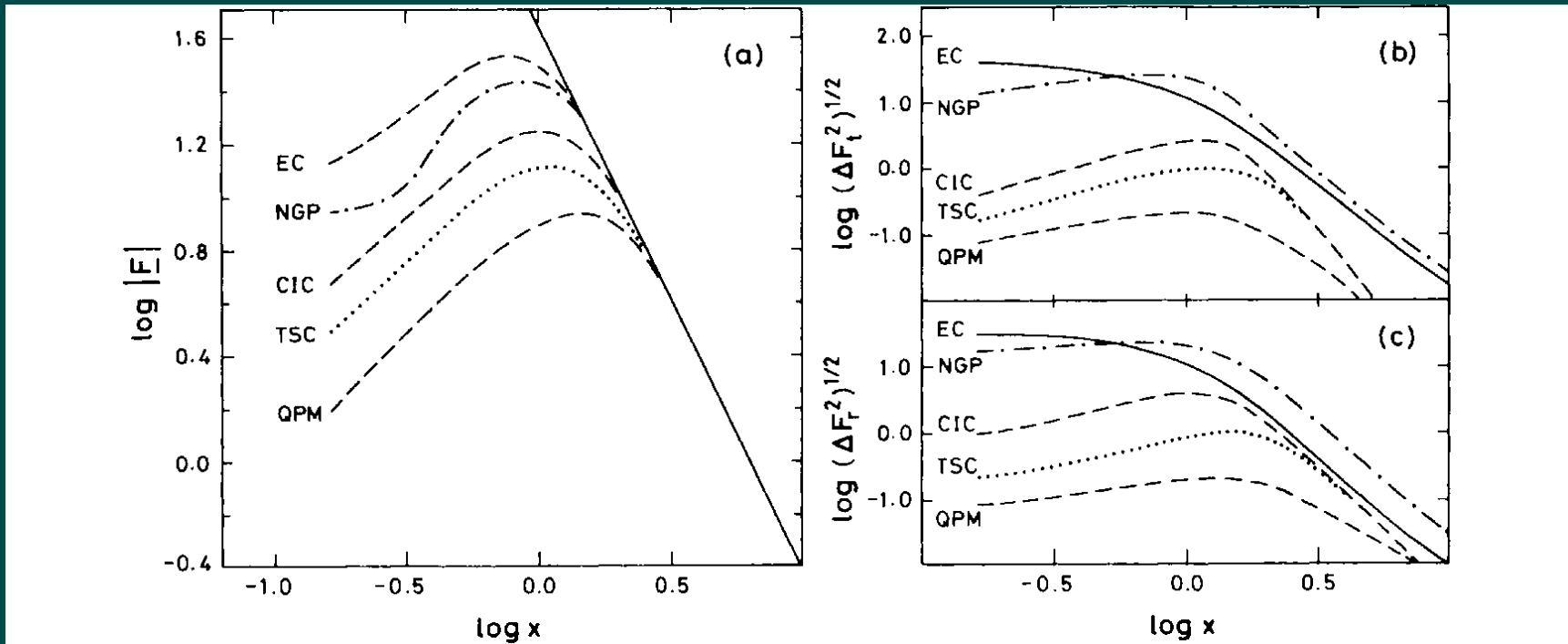
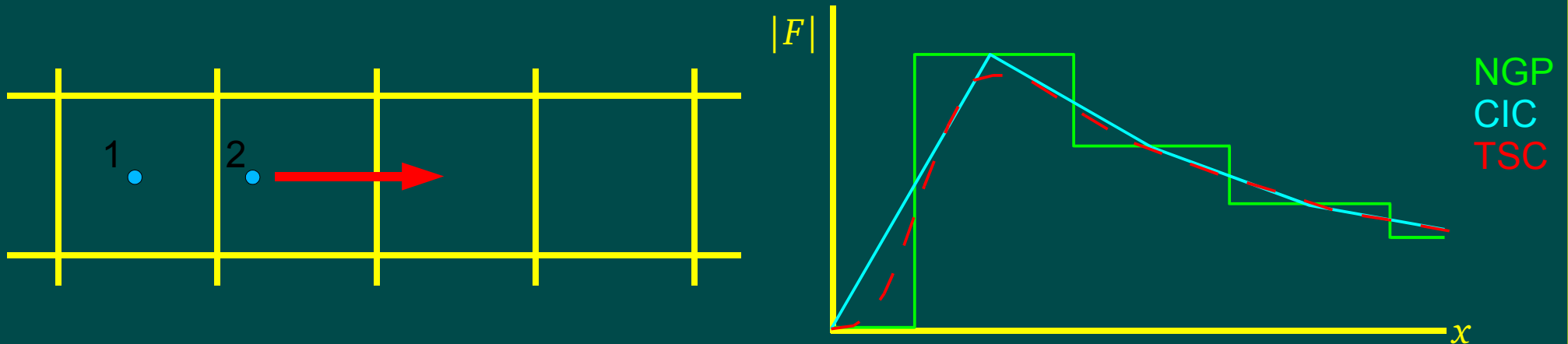
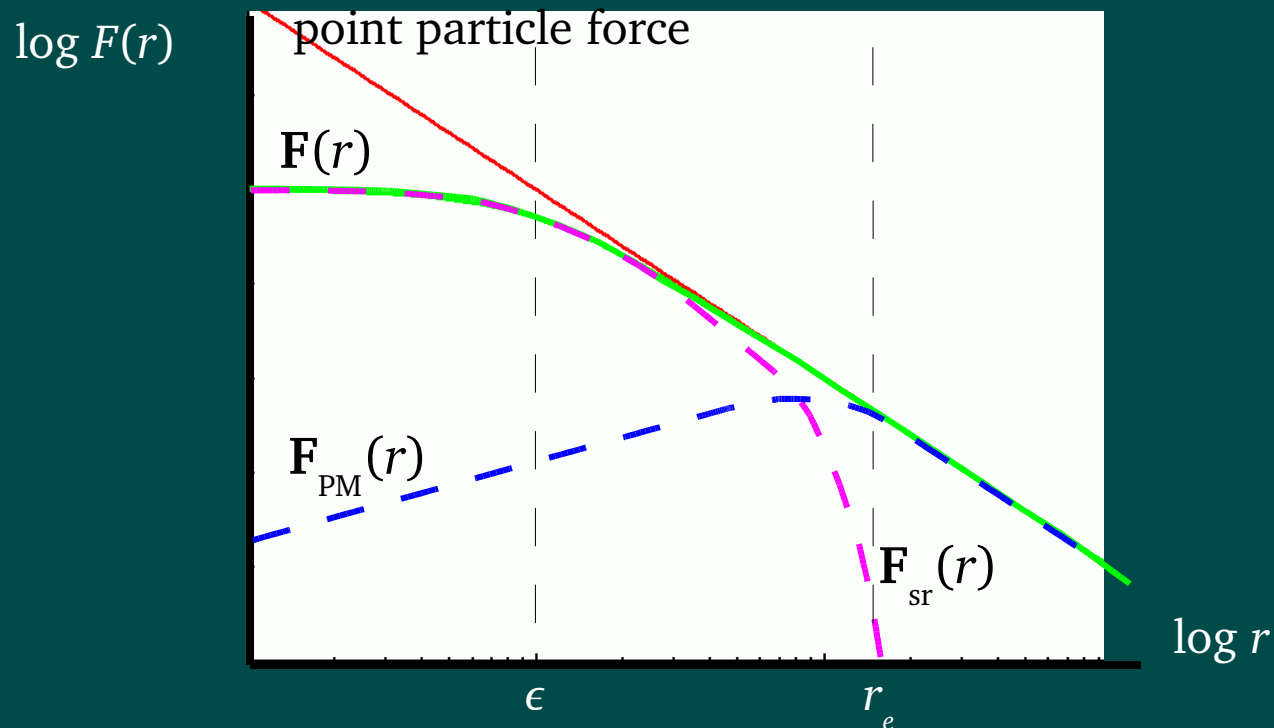


Figure 2 (a) The mean interparticle force as a function of separation in mesh spaces for several particle-mesh schemes. The solid line in (a) shows the unsoftened force. The other panels show the rms fluctuations in the tangential (b) and radial (c) directions [reproduced from Efstathiou et al. (1985), with permission].



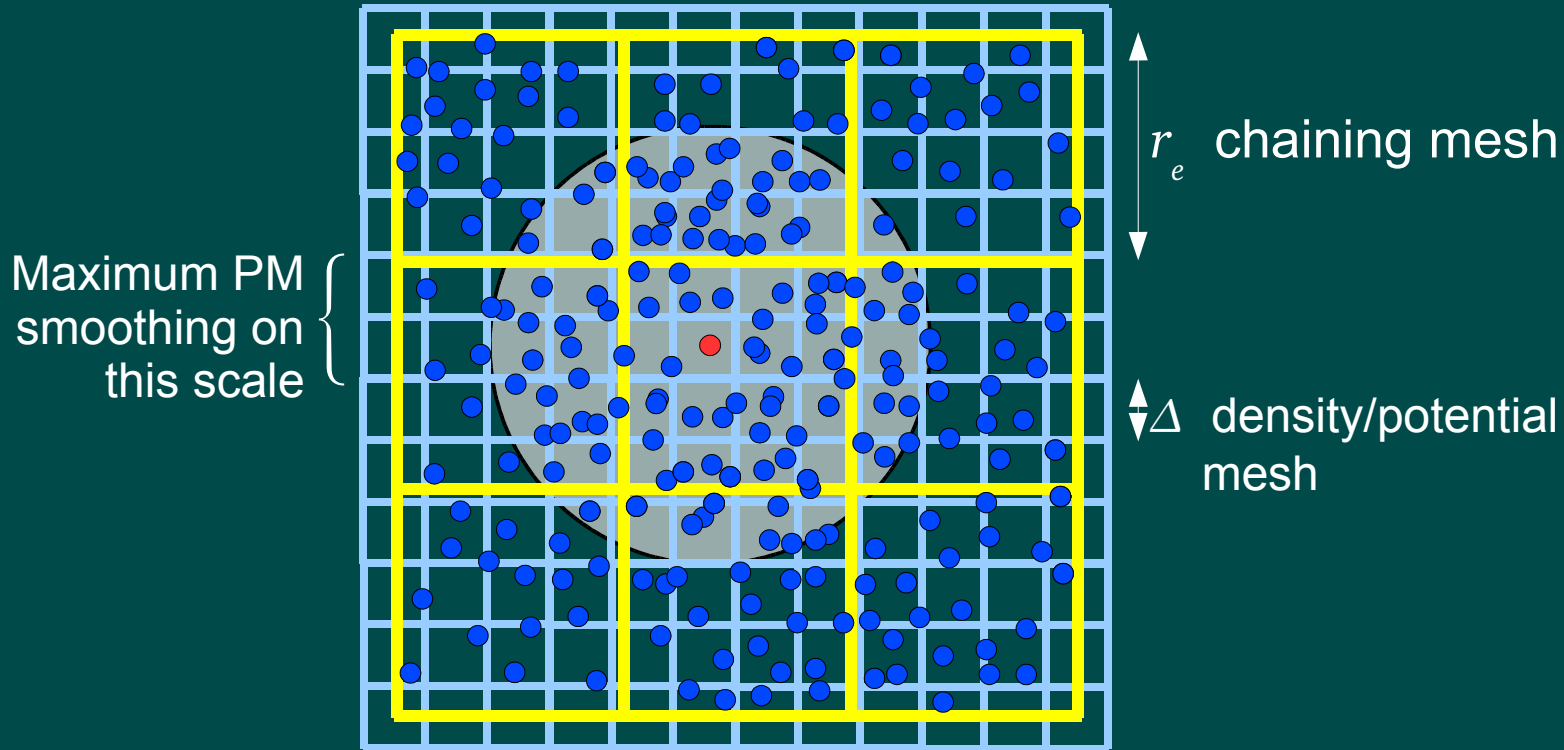
Particle-particle-particle-mesh (P³M)

- Invented in early 1970s by R. W. Hockney and J. W. Eastwood
- Improves force accuracy of PM on small scales without cost of PP
- Procedure:
 1. Compute long-range **mesh force** $\mathbf{F}_{\text{PM}}(r)$ using PM
 2. Desired force $\mathbf{F}(r)$: $\mathbf{F}(r) - \mathbf{F}_{\text{PM}}(r)$ is the **short-range force** $\mathbf{F}_{\text{sr}}(r)$
 3. Total force on each particle is $\mathbf{F}_{\text{PM}}(r) + \sum_{\text{neighbors}} \mathbf{F}_{\text{sr}}(r)$



P³M nearest neighbor search

To find “nearby” particles ($r < r_e$) use a **chaining mesh**:



- Neighbor search restricted to particles lying within the same chaining mesh cell (and immediate neighbors)
- Cost of force computation:

$$\text{CPU time } \tau = \alpha N_p + \beta N_g^3 \ln N_g^3 + \gamma \left(\frac{r_e}{\Delta} \right)^3 \frac{N_p^2}{N_g^3}$$

Mass assignment
Force interpolation

Mesh Poisson solve

Short-range PP force

Treecodes

Developed in mid-1980s independently by A. W. Appel, J. G. Jernigan & D. H. Porter, and J. E. Barnes & P. Hut

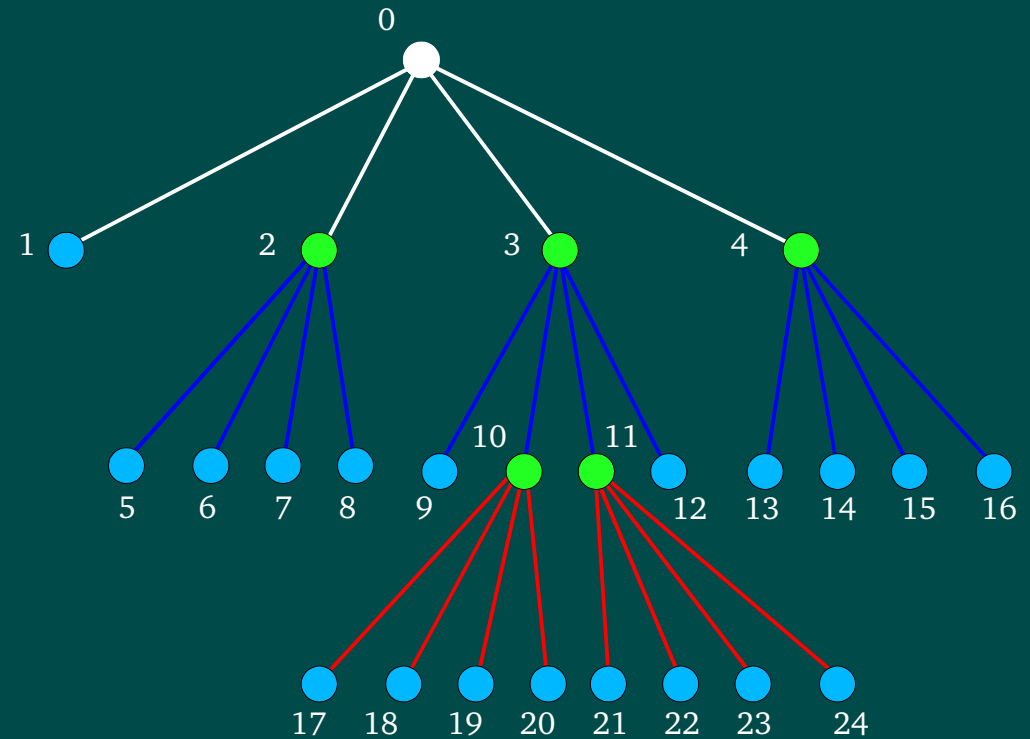
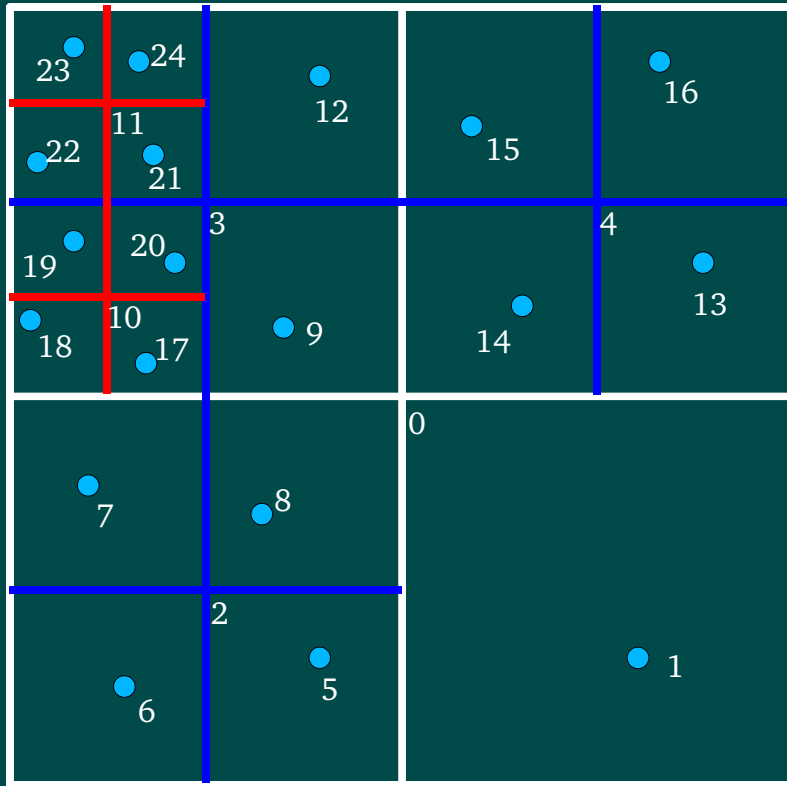
Basic idea:

1. Store particles in a tree data structure. Particles are at *leaves* of tree; *parent nodes* store total mass of children.
2. When the force on a particle is needed, we traverse tree, starting at the *root*.
 - (a) If a node is terminal (leaf node), directly sum the force from the particle at that node.
 - (b) If not, ask: is the monopole (or quadrupole, ...) of the node an adequate approximation to the force from the child particles?
 - (i) If yes, use the approximate force and stop traversing this branch.
 - (ii) If no, traverse the children.

Answer to 2. (b) is determined using a **multipole acceptance criterion (MAC)**.

- Nearby particles get directly summed.
- Distant particles get approximated.

Treecodes



Advantages:

- No grid to limit resolution! (Must introduce force softening explicitly...)
- Scales as $N \ln N$
- Parallelizes extremely well
- Isolated boundaries are natural

Disadvantages:

- Error properties harder to analyze than mesh-based methods
- Periodic boundaries must be introduced via Ewald summation

Multipole acceptance criteria

MACs are usually expressed in terms of an opening angle θ ; # of interactions $\propto \theta^{-3}$

Barnes-Hut MAC

- Use multipole iff $s/r < \theta$
- Typically $0.7 \leq \theta \leq 1.0$



Min-distance MAC

- Use multipole iff $s/r < \theta$
- Does not require knowledge of node contents



The Poisson equation

Integral form:

$$\phi(\mathbf{x}) = -G \iiint d^3 x' \frac{\rho(\mathbf{x}')}{|\mathbf{x} - \mathbf{x}'|} \quad -\frac{G}{|\mathbf{x} - \mathbf{x}'|} = \text{Green's function}$$

$$\nabla \phi(\mathbf{x}) = G \iiint d^3 x' \frac{\mathbf{x} - \mathbf{x}'}{|\mathbf{x} - \mathbf{x}'|^3} \rho(\mathbf{x}') = -\mathbf{g}(\mathbf{x})$$

Differential form:

$$\begin{aligned} \nabla^2 \phi(\mathbf{x}) &= G \iiint d^3 x' \nabla \cdot \left[\frac{\mathbf{x} - \mathbf{x}'}{|\mathbf{x} - \mathbf{x}'|^3} \right] \rho(\mathbf{x}') \\ &= G \iiint d^3 x' \rho(\mathbf{x}') \underbrace{\nabla \cdot \left[\frac{\mathbf{x} - \mathbf{x}'}{|\mathbf{x} - \mathbf{x}'|^3} \right]}_{4\pi \delta(\mathbf{x} - \mathbf{x}')} \\ &= 4\pi G \rho(\mathbf{x}) \end{aligned}$$

Transform methods

General idea:

1. Substitute an integral transform into both sides of equation

$$D_x \phi(\mathbf{x}) = \rho(\mathbf{x})$$

$$\phi(\mathbf{x}) = \int \phi(\mathbf{k}) K(\mathbf{k}, \mathbf{x}) d^3 k \quad \rho(\mathbf{x}) = \int \rho(\mathbf{k}) K(\mathbf{k}, \mathbf{x}) d^3 k$$

$$\phi(\mathbf{k}) D_x K(\mathbf{k}, \mathbf{x}) = \rho(\mathbf{k}) K(\mathbf{k}, \mathbf{x})$$

2. Linear equation does not couple modes in transform space, so we can compute transform of solution directly from transform of source

$$\phi(\mathbf{k}) = \rho(\mathbf{k}) \frac{K(\mathbf{k}, \mathbf{x})}{D_x K(\mathbf{k}, \mathbf{x})} = f(\mathbf{k}) \rho(\mathbf{k})$$

3. Apply inverse transform to obtain the solution in real space

$$\begin{aligned} \phi(\mathbf{x}) &= \int \phi(\mathbf{k}) K(\mathbf{k}, \mathbf{x}) d^3 k = \int f(\mathbf{k}) \rho(\mathbf{k}) K(\mathbf{k}, \mathbf{x}) d^3 k \\ &= \int d^3 k f(\mathbf{k}) K(\mathbf{k}, \mathbf{x}) \int d^3 x' K^*(\mathbf{k}, \mathbf{x}') \rho(\mathbf{x}') \end{aligned}$$

The function $f(\mathbf{k})$ is the transform of the Green's function for the equation.

Transform example

Example: Gravitational Poisson equation, Cartesian geometry, periodic boundaries – Fourier transform

$$K(\mathbf{k}, \mathbf{x}) \equiv \frac{e^{i\mathbf{k}\cdot\mathbf{x}}}{(2\pi)^{3/2}} \quad D_x \equiv \frac{1}{4\pi G} \nabla^2$$

$$D_x K(\mathbf{k}, \mathbf{x}) = \frac{1}{4\pi G} \nabla^2 \left[\frac{e^{i\mathbf{k}\cdot\mathbf{x}}}{(2\pi)^{3/2}} \right] = -\frac{k^2}{4\pi G} K(\mathbf{k}, \mathbf{x})$$

$$f(\mathbf{k}) = \frac{K(\mathbf{k}, \mathbf{x})}{D_x K(\mathbf{k}, \mathbf{x})} = -\frac{4\pi G}{k^2}$$

$$\phi(\mathbf{x}) = -4\pi G \int \frac{\rho(\mathbf{k})}{k^2} e^{i\mathbf{k}\cdot\mathbf{x}} d^3 k$$

Boundary conditions

Transform basis functions must satisfy the boundary conditions:

Dirichlet ($\phi=0$): Fourier sine transform $K(\mathbf{K}, \mathbf{x}) = \sin\left(\frac{1}{2} \mathbf{k} \cdot \mathbf{x}\right)$

Neumann ($\mathbf{n} \cdot \nabla \phi = 0$): Fourier cosine transform $K(\mathbf{K}, \mathbf{x}) = \cos\left(\frac{1}{2} \mathbf{k} \cdot \mathbf{x}\right)$

Given-value: use Dirichlet with modified source function
e.g., for $\nabla^2 \phi = \rho$ we have

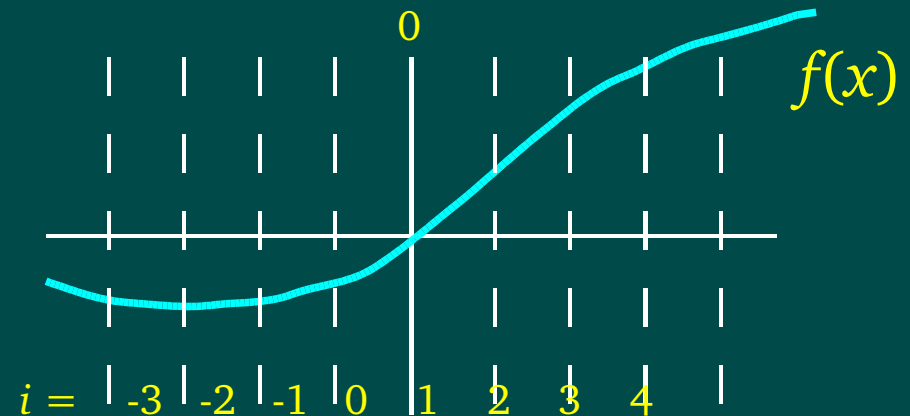
$$\phi_0 = \phi(0) - \frac{1}{2} \Delta x \phi'(0) + O(\Delta x^2)$$

$$\phi_1 = \phi(0) + \frac{1}{2} \Delta x \phi'(0) + O(\Delta x^2)$$

$$\Rightarrow \phi_0 = 2\phi(0) - \phi_1$$

so use

$$\rho'_i = \begin{cases} \rho_i - \frac{2\phi(0)}{\Delta x^2} & i=1 \\ \rho_i & i>1 \end{cases}$$



Fourier transform

Direct Fourier Transform (DFT) in 1D:

1. Compute DFT of density field – $O(N^2)$ operations

$$\hat{\rho}_p = \frac{1}{(2\pi)^{3/2}} \sum_{q=0}^{N-1} e^{2\pi p q i} \rho_q \quad p=0 \dots N-1$$

2. Apply Green's function to obtain transform of potential – $O(N)$ operations

$$\hat{\phi}_p = -\frac{4\pi G}{k_p^2} \hat{\rho}_p = -\frac{GL^2}{\pi p^2} \hat{\rho}_p \quad p=0 \dots N-1$$

3. Compute inverse DFT to obtain potential field – $O(N^2)$ operations

$$\phi_q = \frac{1}{(2\pi)^{3/2}} \sum_{p=0}^{N-1} e^{-2\pi p q i} \hat{\phi}_p \quad q=0 \dots N-1$$

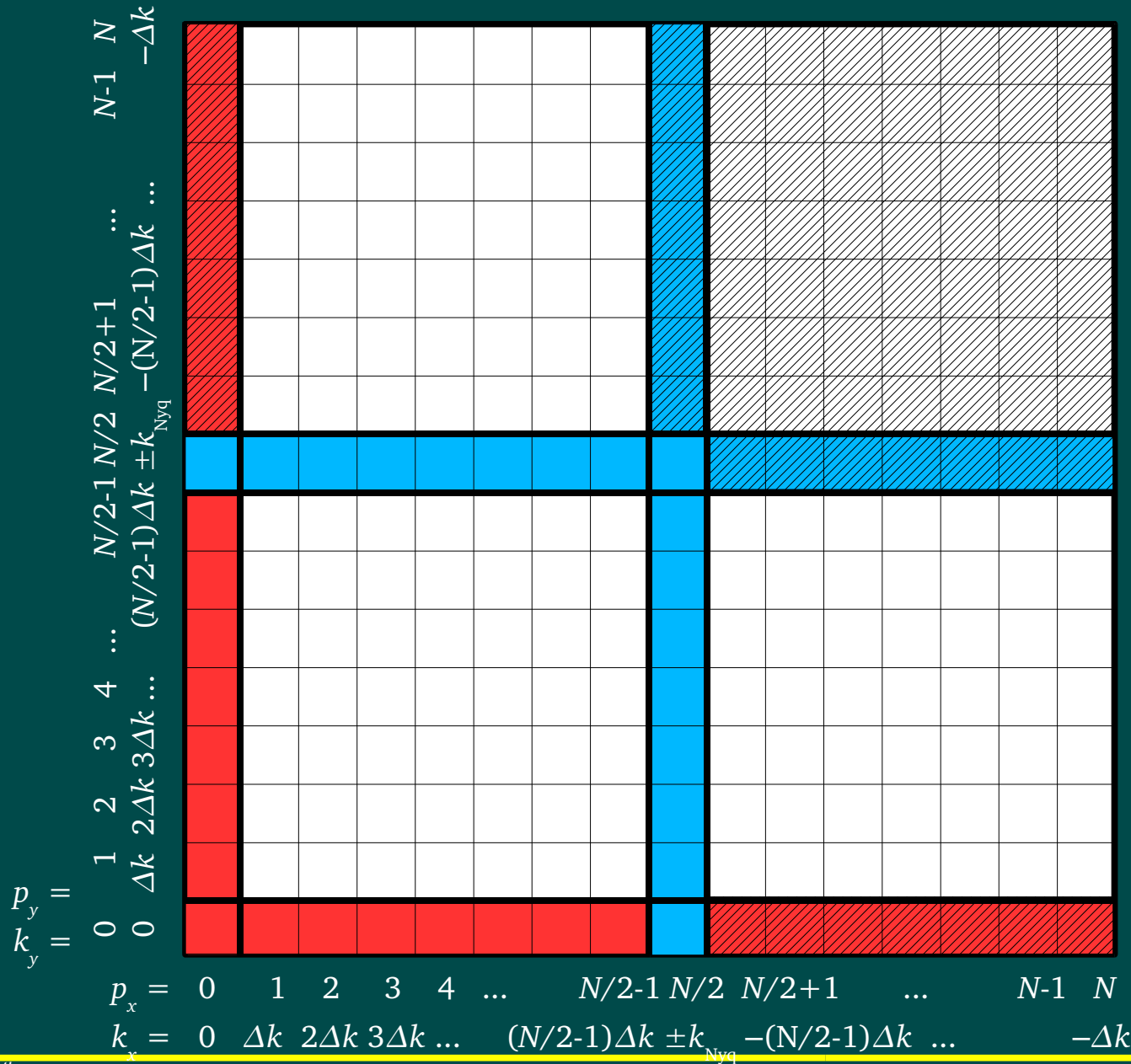
Fast Fourier Transform (FFT) replaces steps 1 and 3 with $O(N \ln N)$ operations

Example: $N = 100$

$$\frac{N^2}{N \ln N} \approx \frac{10^4}{100 \cdot 4.61} \approx 21.7$$

Multidimensional FFT

Array ordering of 2D FFT results; 3D similar (Press et al. routines)



$$\hat{\rho}(-\mathbf{k}) = \hat{\rho}(\mathbf{k})^*$$

for real-valued $\rho(\mathbf{x})$;
so hatched zones are
not independent.

Some FFTs exploit this
fact to reduce storage
requirements.

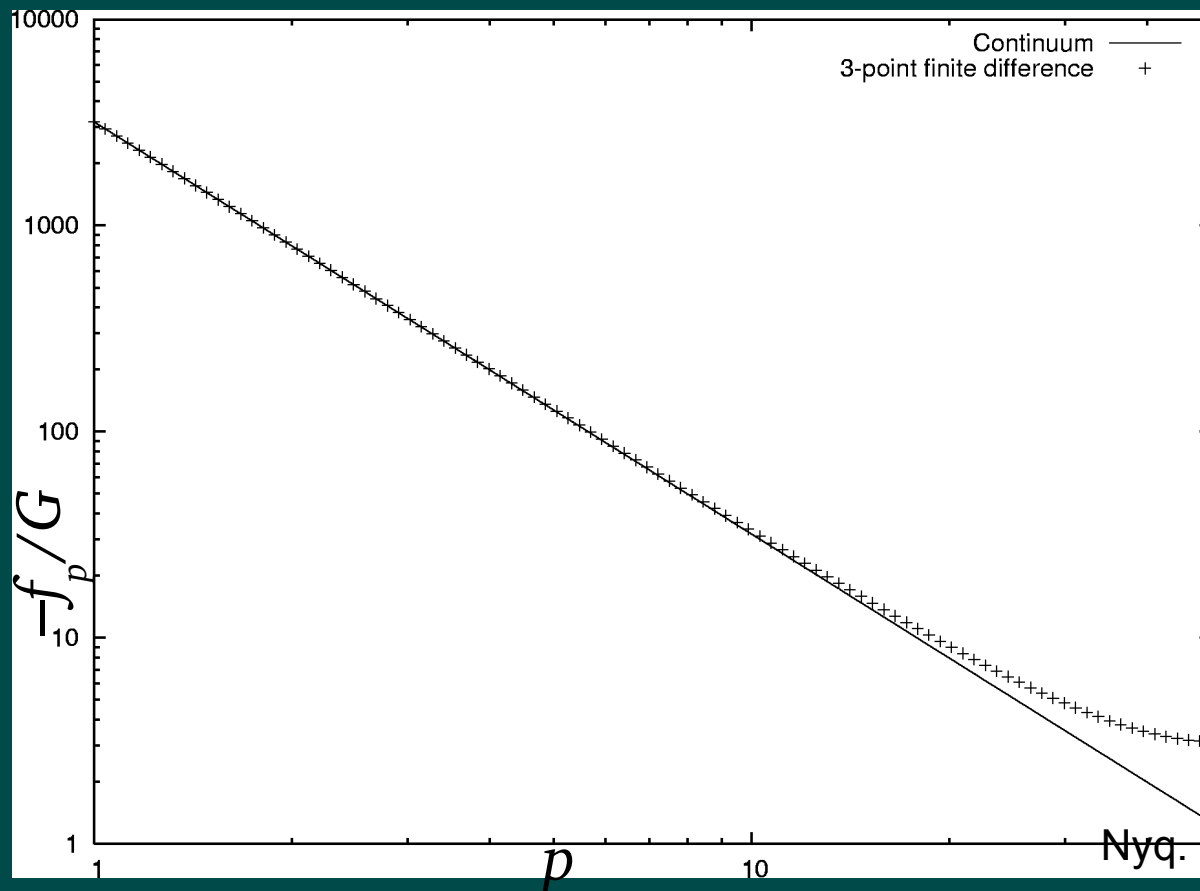
Knowledge of the
ordering is needed
when multiplying by the
Green's function
transform.

Green's function

Continuum Green's function $f_{p_1, p_2, p_3} = f(\mathbf{k}_{p_1, p_2, p_3}) = -\frac{N^2 G}{\pi} \frac{1}{p_1^2 + p_2^2 + p_3^2}$

7-point (second order) finite difference

$$f_{p_1, p_2, p_3} = -\pi G \left[\sin^2\left(\frac{\pi p_1}{N}\right) + \sin^2\left(\frac{\pi p_2}{N}\right) + \sin^2\left(\frac{\pi p_3}{N}\right) \right]^{-1}$$



Multipole methods

For nearly spherical matter distributions, we can obtain a good approximation to the potential in $O(N)$ steps using multipoles.

Same idea as with Fourier transforms – different basis; but we truncate expansion.

Note that

$$\frac{1}{|\mathbf{x} - \mathbf{x}'|} = 4\pi \sum_{l=0}^{\infty} \sum_{m=-l}^l \frac{1}{2l+1} \frac{r_{<}^l}{r_{>}^{l+1}} Y_{lm}^*(\theta', \varphi') Y_{lm}(\theta, \varphi)$$

where

$$r_{<} \equiv \min[|\mathbf{x}|, |\mathbf{x}'|]$$
$$r_{>} \equiv \max[|\mathbf{x}|, |\mathbf{x}'|]$$

The Y_{lm} are spherical harmonics:

$$Y_{lm}(\theta, \varphi) \equiv (-1)^m \sqrt{\frac{2l+1}{4\pi} \frac{(l-m)!}{(l+m)!}} P_{lm}(\cos\theta) e^{im\varphi}$$

P_{lm} are the Legendre polynomials.

Multipole methods – 2

Using the integral form of the Poisson equation, we obtain

$$\phi(\mathbf{x}) = -G \sum_{l=0}^{\infty} \sum_{m=-l}^l \frac{1}{2l+1} Y_{lm}(\theta, \varphi) \left[r^l \int_{r < r'} d^3 x' \frac{\rho(\mathbf{x}')}{r'^{l+1}} Y_{lm}^*(\theta', \varphi') + \frac{1}{r^{l+1}} \int_{r > r'} d^3 x' \rho(\mathbf{x}') r'^l Y_{lm}^*(\theta', \varphi') \right]$$

Note that

$$\sum_{m=-l}^l Y_{lm}^*(\theta', \varphi') Y_{lm}(\theta, \varphi) = \frac{2l+1}{4\pi} \left[P_{l0}(\cos \theta) P_{l0}(\cos \theta') + 2 \sum_{m=1}^l \frac{(l-m)!}{(l+m)!} P_{lm}(\cos \theta) P_{lm}(\cos \theta') \cos(m(\varphi - \varphi')) \right]$$

We can use a trig identity on the last cosine.

Multipole methods – 3

$$\phi(\mathbf{x}) = -G \sum_{l=0}^{\infty} P_{l0}(\cos \theta) \left[r^l \mu_{l0}^{eo}(r) + \frac{1}{r^{l+1}} \mu_{l0}^{ei}(r) \right] -$$

$$2G \sum_{l=1}^{\infty} \sum_{m=1}^l P_{lm}(\cos \theta) \left[(r^l \cos m \varphi) \mu_{lm}^{eo}(r) + (r^l \sin m \varphi) \mu_{lm}^{oo}(r) + \right.$$

$$\left. \frac{\cos m \varphi}{r^{l+1}} \mu_{lm}^{ei}(r) + \frac{\sin m \varphi}{r^{l+1}} \mu_{lm}^{oi}(r) \right]$$

where the even/odd, inner/outer moments of the density are given by

$$\mu_{lm}^{ei}(r) \equiv \frac{(l-m)!}{(l+m)!} \int_{r>r'} d^3 x' r'^l \rho(\mathbf{x}') P_{lm}(\cos \theta') \cos m \varphi'$$

$$\mu_{lm}^{oi}(r) \equiv \frac{(l-m)!}{(l+m)!} \int_{r>r'} d^3 x' r'^l \rho(\mathbf{x}') P_{lm}(\cos \theta') \sin m \varphi'$$

$$\mu_{lm}^{eo}(r) \equiv \frac{(l-m)!}{(l+m)!} \int_{r<r'} d^3 x' r'^{-(l+1)} \rho(\mathbf{x}') P_{lm}(\cos \theta') \cos m \varphi'$$

$$\mu_{lm}^{oo}(r) \equiv \frac{(l-m)!}{(l+m)!} \int_{r<r'} d^3 x' r'^{-(l+1)} \rho(\mathbf{x}') P_{lm}(\cos \theta') \sin m \varphi'$$

Evaluate by any appropriate $O(N)$ quadrature method.

Relaxation methods

Relaxation methods treat the solution to an elliptic problem as the steady-state solution to a diffusion problem:

$$\nabla^2 \phi = \rho \quad \rightarrow \quad \frac{\partial \phi}{\partial t} = \nabla^2 \phi - \rho$$

Here the “time coordinate” is really an iteration coordinate.

For example, FTCS differencing of this equation in 1D gives:

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \frac{\phi_{i+1}^n - 2\phi_i^n + \phi_{i-1}^n}{\Delta x^2} - \rho_i$$

$$\phi_i^{n+1} = \phi_i^n + \frac{\Delta t}{\Delta x^2} [\phi_{i+1}^n - 2\phi_i^n + \phi_{i-1}^n] - \Delta t \rho_i$$

CFL-like stability criterion gives $\Delta t = \Delta x^2/2$ (1D), $\Delta x^2/4$ (2D):

$$\phi_i^{n+1} = \frac{1}{2} [\phi_{i+1}^n + \phi_{i-1}^n] - \frac{\Delta x^2}{2} \rho_i$$

This is the **Jacobi method**.

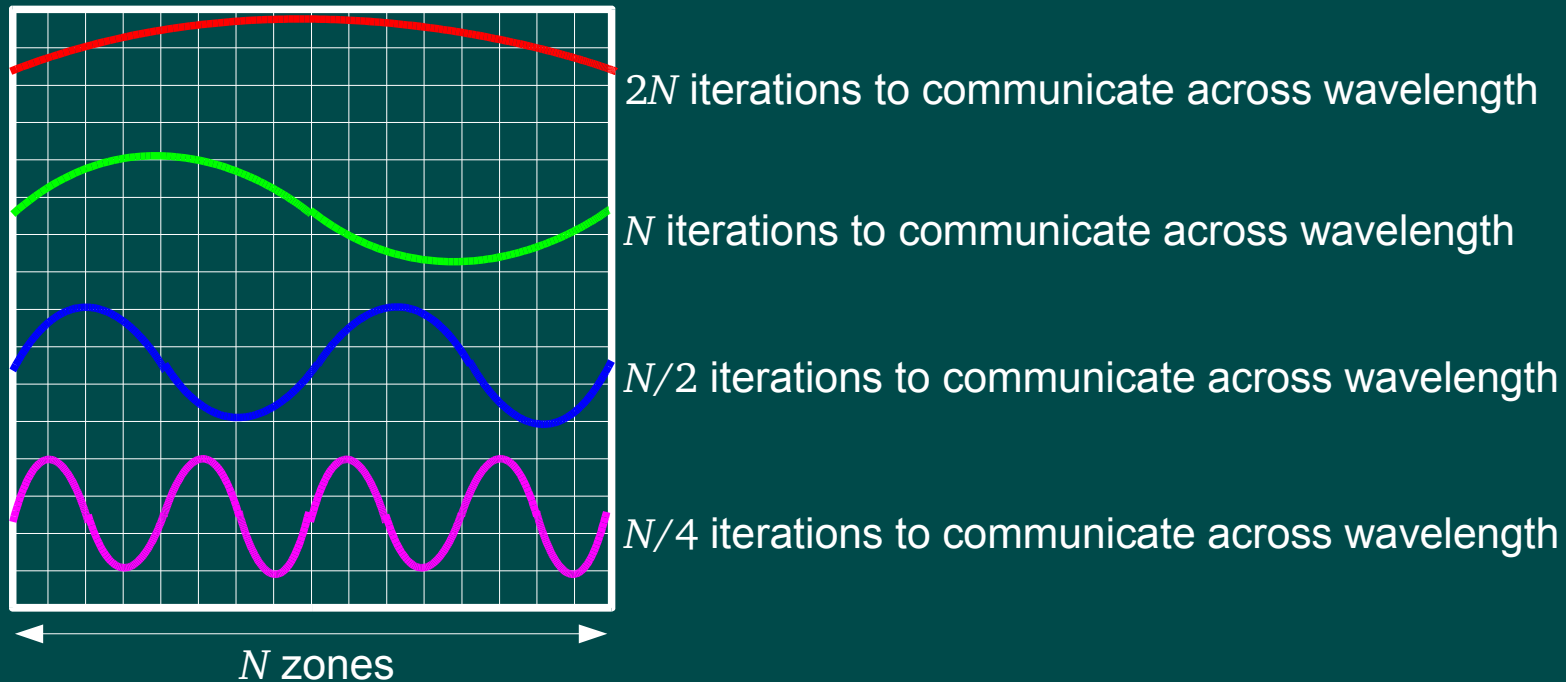
Relaxation methods

The **Gauss-Seidel method** is similar, but uses updated quantities when available:

$$\phi_i^{n+1} = \frac{1}{2} \left[\phi_{i+1}^n + \phi_{i-1}^{n+1} \right] - \frac{\Delta x^2}{2} \rho_i$$

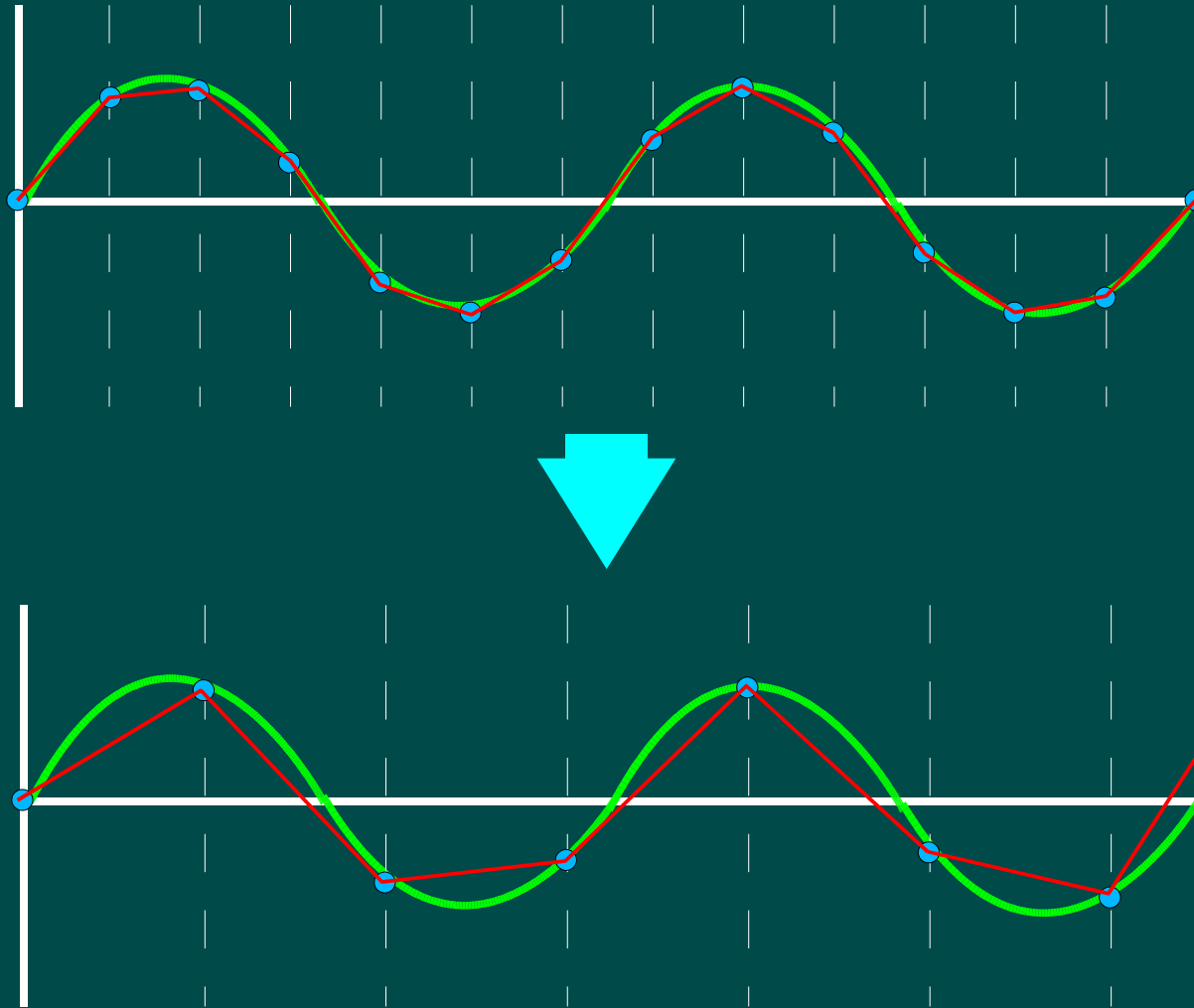
Both of these methods are very slow: to reduce overall error by 10^p requires of order pN iterations, each of which costs N operations: $O(N^2)$ overall.

Why so slow? Difference operator is *local* – so long-wavelength errors require $\sim N$ iterations for information to propagate across grid.



Multigrid methods – basic idea

On a coarser mesh, a given error mode appears to be “higher frequency:”



Want to get long- and short-wavelength modes to converge at same rate.

Multigrid – two-grid iteration

1. Start with the Poisson equation discretized on a mesh with spacing h .

$$\frac{\partial^2 \phi}{\partial x^2} = \rho \quad \rightarrow \quad \frac{\phi_{i-1} - 2\phi_i + \phi_{i+1}}{h^2} = \rho_i \quad \rightarrow \quad D_{[h]} \phi_{[h]} = \rho_{[h]}$$

2. Given a solution guess $\tilde{\phi}_{[h]}$, apply a few iterations of a relaxation scheme on mesh h to reduce short-wavelength noise.

$$\tilde{\phi}_{[h]} \rightarrow S_{[h]} \tilde{\phi}_{[h]}$$

3. Compute the **residual** $r_{[h]}$ of the smoothed guess. The solution to the original (linear) equation with the residual as source term is the **correction**.

$$D_{[h]}(\tilde{\phi}_{[h]} + c_{[h]}) = \rho_{[h]} \quad D_{[h]}c_{[h]} = \rho_{[h]} - D_{[h]}\tilde{\phi}_{[h]} \equiv r_{[h]}$$

4. **Restrict** the residual to a mesh with spacing $2h$.

$$r_{[2h]} = R_{[2h]}^{[h]} r_{[h]}$$

Multigrid – two-grid iteration

5. Solve the residual equation on mesh $2h$ exactly using some method.

$$D_{[2h]}c_{[2h]} = r_{[2h]}$$

6. **Prolongate** the correction from mesh $2h$ to mesh h .

$$c_{[h]} = P_{[h]}^{[2h]} c_{[2h]}$$

7. Apply the correction to the initial solution guess.

$$\tilde{\phi}_{[h]} \rightarrow \tilde{\phi}_{[h]} + c_{[h]}$$

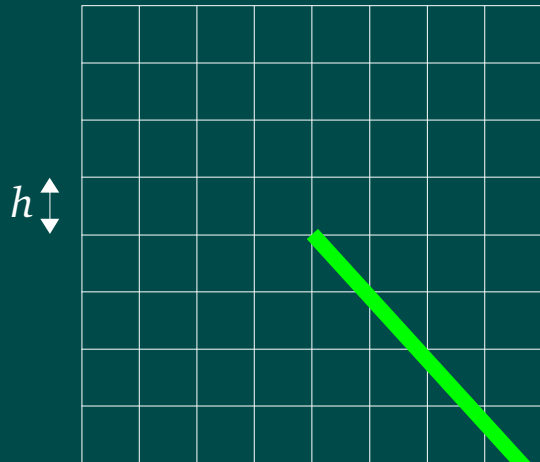
8. Apply a few iterations of the relaxation operator to the new solution.

$$\tilde{\phi}_{[h]} \rightarrow S_{[h]} \tilde{\phi}_{[h]}$$

9. Repeat if necessary.

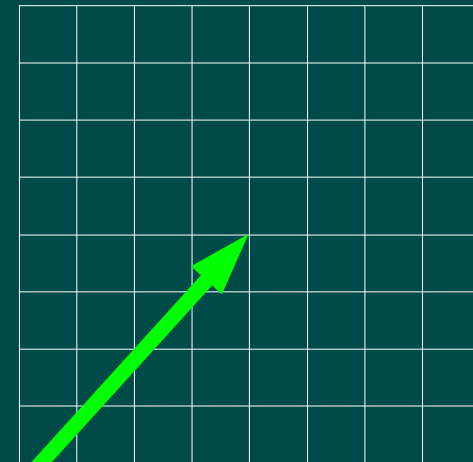
Smooth $\tilde{\phi}_{[h]} \rightarrow S_{[h]} \tilde{\phi}_{[h]}$

Residual $r_{[h]} = \rho_{[h]} - D_{[h]} \tilde{\phi}_{[h]}$

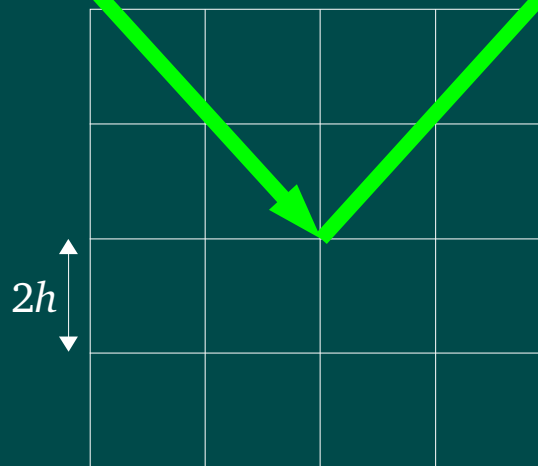


Correct $\tilde{\phi}_{[h]} \rightarrow \tilde{\phi}_{[h]} + c_{[h]}$

Smooth $\tilde{\phi}_{[h]} \rightarrow S_{[h]} \tilde{\phi}_{[h]}$



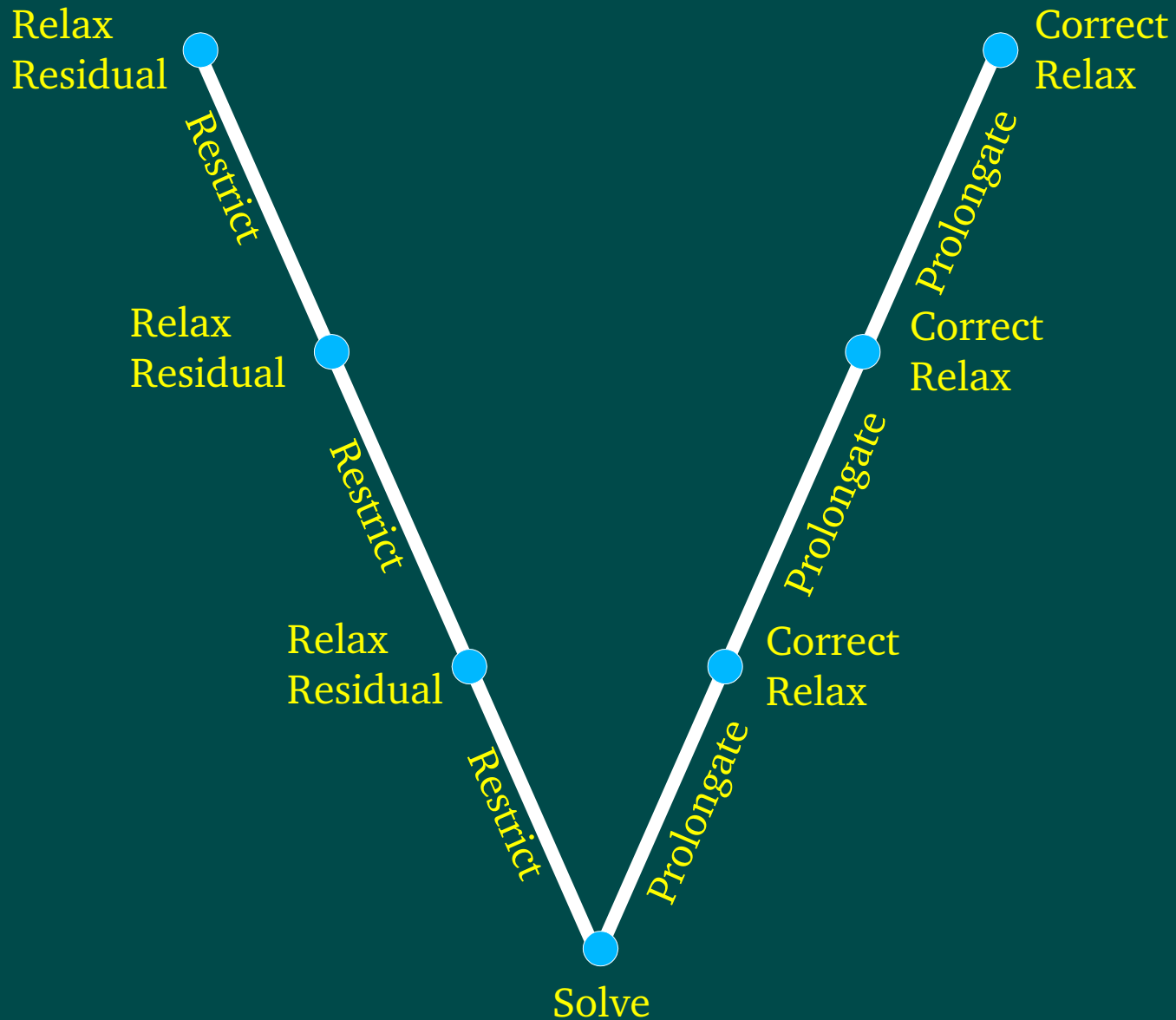
Restrict $r_{[2h]} = R_{[2h]}^{[h]} r_{[h]}$



Prolongate $c_{[h]} = P_{[h]}^{[2h]} c_{[2h]}$

Solve $D_{[2h]} c_{[2h]} = r_{[2h]}$

Multigrid methods – V-cycle



Coupling gravity to hydrodynamics

1. For PPM, linear corrections to Riemann solver input states must be made to account for source terms (gravity, non-Cartesian coordinates, etc.)
2. As a source term in the Euler (momentum) and energy equations. Momentum equation term is usually written

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot [\rho \mathbf{u} \mathbf{u} + P \mathbf{I}] = \rho \mathbf{g}$$

but can be converted into a conservative tensor flux if the gravitational field is generated by the matter (instead of being externally imposed):

$$\begin{aligned}\nabla \cdot \mathbf{g} &= -\nabla^2 \phi = -4\pi G \rho \\ \rho \mathbf{g} &= -\frac{1}{4\pi G} \mathbf{g} \nabla \cdot \mathbf{g} \\ &= -\frac{1}{4\pi G} [\nabla \cdot (\mathbf{g} \mathbf{g}) - (\mathbf{g} \cdot \nabla) \mathbf{g}] \\ &= -\frac{1}{4\pi G} \left[\nabla \cdot (\mathbf{g} \mathbf{g}) - \frac{1}{2} \nabla \cdot (|\mathbf{g}|^2 \mathbf{I}) \right] \\ &= \nabla \cdot \mathbf{G}\end{aligned}$$

Coupling gravity to hydrodynamics

Define the gravitational stress tensor as

$$\mathbf{G} \equiv -\frac{1}{4\pi G} \left[\mathbf{g} \mathbf{g} - \frac{1}{2} |\mathbf{g}|^2 \mathbf{I} \right]$$

The energy equation, however, cannot be written in conservative form:

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot [(\rho E + P) \mathbf{u}] = \rho \mathbf{u} \cdot \mathbf{g}$$

$$\begin{aligned} \rho \mathbf{u} \cdot \mathbf{g} &= -\rho \mathbf{u} \cdot \nabla \phi \\ &= -\nabla \cdot (\rho \mathbf{u} \phi) - \frac{\partial \rho \phi}{\partial t} + \rho \frac{\partial \phi}{\partial t} \end{aligned}$$

If we redefine E as

$$E \rightarrow E' \equiv \frac{1}{2} |\mathbf{u}|^2 + \varepsilon + \phi$$

we can write

$$\frac{\partial \rho E'}{\partial t} + \nabla \cdot [(\rho E' + P) \mathbf{u}] = \rho \frac{\partial \phi}{\partial t}$$

but the time derivative of potential cannot be turned into a flux. Because of the long-range nature of gravity, potential energy cannot be “advected.”