Faculty of Science

Computational Astrophysics

4a. Selfgravity: Solving the Poisson Equation

Numerical Methods in Astrophysics Chap 7

Troels Haugbølle

Niels Bohr Institute

University of Copenhagen

How do we calculate the force of self gravity ?

- Direct methods
  - Direct Force Evaluation using Newton's Law
  - FFT: Fourier transform $\nabla^2\Phi = 4\pi G\rho$
- Iterative methods to solve $\nabla^2\Phi = 4\pi G\rho$
  - Jacobi, Gauss-Seidel, and SOR
  - Multigrid

Direct Force Evaluation – chapter 7.1

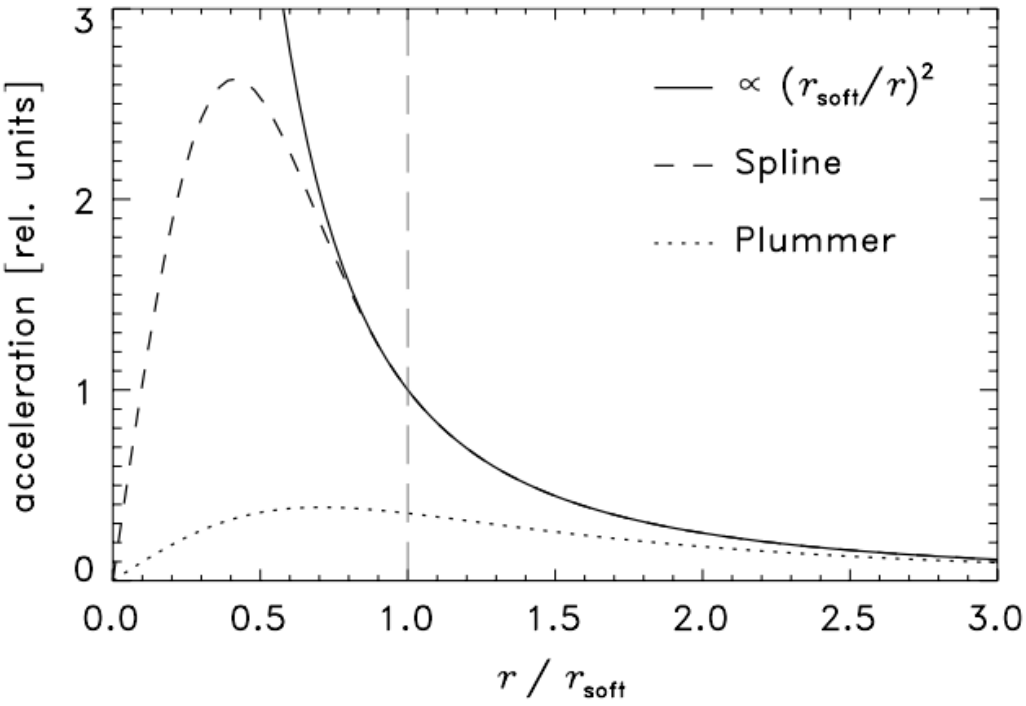
- Use Newton's law between particles or fluid elements
- Problem: scales like $O(N^2)$ – impractical above 10^4 particles
- Advantage: Simple and grid independent. Adaptive
- Gravity is linear: Direct force calculation can be used with advantage if single objects dominate the potential
  - Stars in a star forming cloud
  - Central object: AGN in a galaxy, star in a proto-planetary disk, etc
- Caveat: Forces / acceleration can become very large for close encounters. Solutions:
  - Use sub-cycling in time for close encounter (e.g. *nbody1-7* codes)
  - Use a smoothening of the force at small distances:

$$\mathbf{F} = G\frac{M_1 M_2}{(r_{12}+\varepsilon)^2}\frac{\mathbf{r}_{12}}{r_{12}} \quad \text{(Plummer softening; } \varepsilon \text{ plummer radius)}$$

$$\mathbf{F} = GM_1M_2\mathbf{g}(\mathbf{r}_{12}) \text{ (Cubic spline softening, Price \& Monaghan 2007)}$$

Direct Force Evaluation

- Use Newton's law between pa
- Problem: scales like O(N²) – i
- Advantage: Simple and grid i
- Gravity is linear: Direct force if single objects dominate the
  - Stars in a star forming clo
  - Central object: AGN in a g etc



- Caveat: Forces / acceleration can become very large for close encounters. Solutions:
  - Use sub-cycling in time for close encounter
  - Use a smoothening of the force at small distances:

$$\mathbf{F} = G\frac{M_1 M_2}{(r_{12}+\varepsilon)^2}\frac{\mathbf{r}_{12}}{r_{12}}$$ (Plummer softening; ε plummer radius)

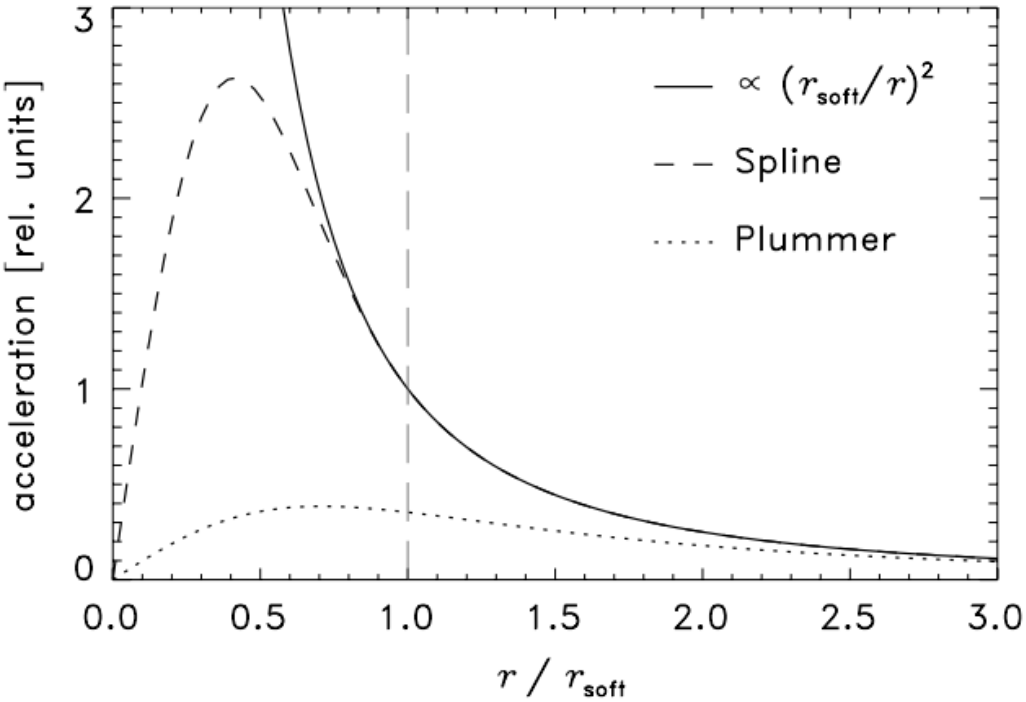
$$\mathbf{F} = GM_1M_2\mathbf{g}(\mathbf{r}_{12})$$ (Cubic spline softening, Price & Monaghan 2007)

Direct Force Evaluation

- Use Newton's law between pa
- Caveat: Forces / acceleration encounters. Solutions:
  - Use sub-cycling in time for
  - Use a smoothening of the

$$\mathbf{F} = G\frac{M_1 M_2}{(r_{12}+\varepsilon)^2}\frac{\mathbf{r}_{12}}{r_{12}}$$ (Plummer softeni

$$\mathbf{F} = GM_1M_2\mathbf{g}(\mathbf{r}_{12})$$ (Cubic spline softer



$$\mathbf{g}(\mathbf{r}) = \begin{cases} \frac{4}{r_{soft}^2}\left[\frac{8}{3}\left(\frac{r}{r_{soft}}\right) - \frac{48}{5}\left(\frac{r}{r_{soft}}\right)^3 + 8\left(\frac{r}{r_{soft}}\right)\right]\frac{\mathbf{r}}{r} & \text{for } 0 \le \frac{r}{r_{soft}} < \frac{1}{2} \\ \frac{4}{r_{soft}^2}\left[\frac{16}{3}\left(\frac{r}{r_{soft}}\right) - 12\left(\frac{r}{r_{soft}}\right)^2 + \frac{48}{5}\left(\frac{r}{r_{soft}}\right)^3 - \frac{8}{3}\left(\frac{r}{r_{soft}}\right)^4 - \frac{1}{60}\left(\frac{r}{r_{soft}}\right)^{-2}\right]\frac{\mathbf{r}}{r} & \text{for } \frac{1}{2} \le \frac{r}{r_{soft}} < 1 \\ \frac{\mathbf{r}}{r^3} & \text{for } \frac{r}{r_{soft}} \ge 1 \end{cases}$$

Direct Force Evaluation: Fourier Transform

- Fast Fourier Transform of the Poisson equation
  - Taking the Fourier transform we can find the potential as

$$\nabla^2\Phi = 4\pi G\rho \rightarrow -k^2\Phi_k = 4\pi G\rho_k$$

  - Even though it involves a Forward FFT for the density and a backward FFT for the potential, it can be competitive on regular meshes since it scales like O(N log(N))
  - Works well for periodic and vacuum boundary conditions
  - But what about non-trivial boundary conditions and adaptive meshes ?
  - The focus on multi-scale problems is making FFT as a gravity solver less relevant today

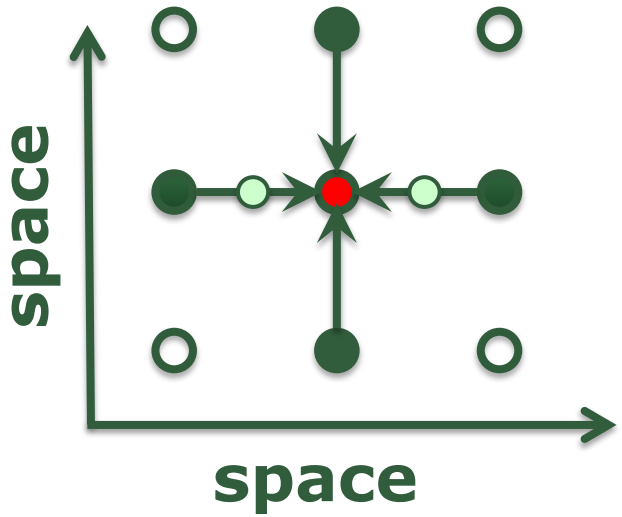
Iterative methods – chapter 7.2

- Solve the Poisson equation using a ***discrete representation of the Laplace operator***

$$\nabla^2\Phi = 4\pi G\rho$$

- The Laplace operator can be discretized as

$$\partial^2\Phi_{i,j,k} / \partial x^2 = (\Phi_{i-1,j,k} - 2\Phi_{i,j,k} + \Phi_{i+1,j,k}) / \Delta x^2$$



- Works with arbitrary boundaries

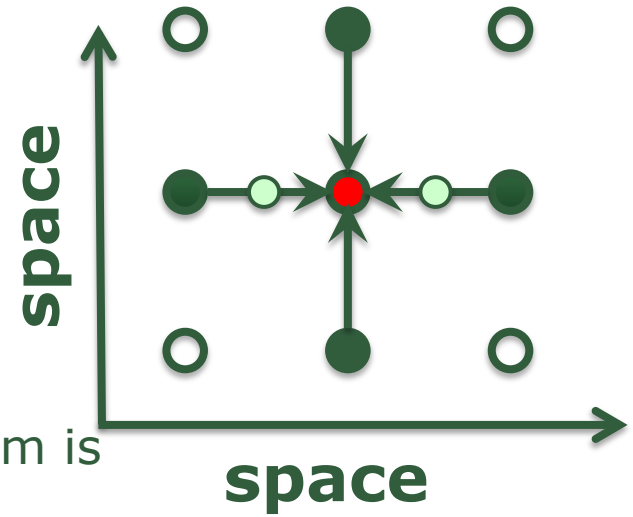
- This couples all points in the domain together and could in principle be solved as a linear equation system, but alternating components is bad for linear stability above ~100 elements (or a 5x5x5 grid!)

Iterative methods – as a time integration

- Solve the Poisson equation using a ***discrete representation of the Laplace operator***, assuming it is the ***steady state solution*** of

$$\partial\Phi / \partial T = \nabla^2\Phi - 4\pi G\rho$$

where T is a "pseudo time".



- Inserting with a simple time integration gives

$$(\Phi^{n+1}_i - \Phi_i)/\Delta T = (\Phi_{i-1} - 2\Phi_i + \Phi_{i+1})/\Delta x^2 - 4\pi G\rho$$

- The Courant condition for this diffusion problem is

$$\Delta T \leq \Delta x^2 / (2\ D)$$

where D = 1, 2, 3 for 1D, 2D, and 3D problems.

- Inserting this timestep gives (for 1D problem)

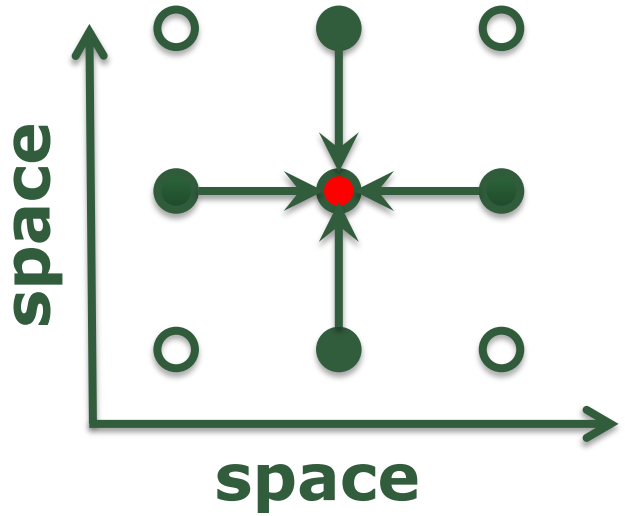
$$2\Phi^{n+1}_i /\Delta x^2 - 2\Phi_i/\Delta x^2 = (\Phi_{i-1} - 2\Phi_i + \Phi_{i+1})/\Delta x^2 - 4\pi G\rho$$

Iterative methods I – Jacobi

- ***Jacobi Iterations:*** Take advantage of the linear nature of the equation:

1. Make a guess for the potential Φ
2. Assume this is the right solution, update iteratively as average of neighbors (assuming $\Delta x = \Delta y$)

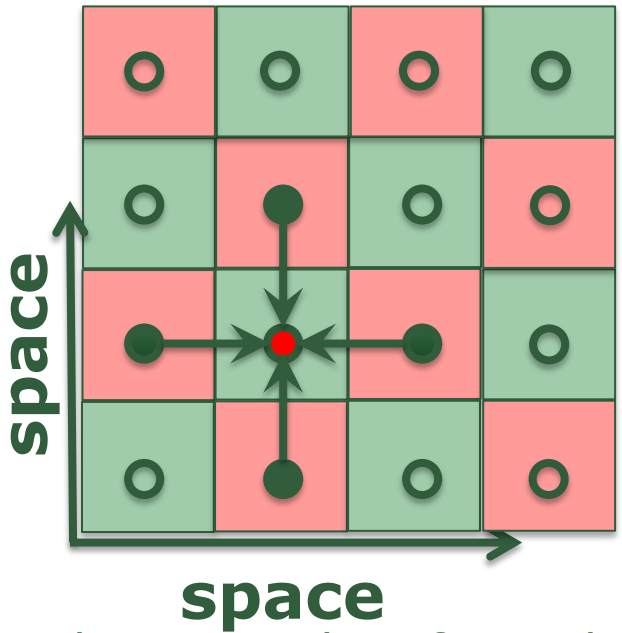
$$6\,\Phi^{n+1}_{i,j,k} = \Phi^{n}_{i-1,j,k} + \Phi^{n}_{i+1,j,k} + \Phi^{n}_{i,j-1,k} + \Phi^{n}_{i,j+1,k} + \Phi^{n}_{i,j,k-1} + \Phi^{n}_{i,j,k+1} - 4\pi G\rho\,\Delta x^2$$



- ***This is equivalent to solving the diffusion equation***
- Problem is it only updates the shortest (grid-scale) wavelengths. Therefore, the method is sensitive to initial guess for the large-scale modes, and converges very slowly – iterations scale like N^2

Iterative methods II – Gauss-Seidel

- ***Gauss-Seidel:*** Observe points are coupled as a checkerboard



- Solve first for green mesh points, then for pink. Effectively doubles the convergence rate. Method is ***in-place in memory. number of grid-cells has to be even, if boundary is periodic***

- Convergence still very bad: Typically, $(N^3)^2$ iterations for a N^3 mesh

- Jacobi and GS methods mostly educational or used for correcting an almost good enough solution.

Iterative methods III – SOR

- ***Succesive Over Relaxation:*** If at every iteration we are moving towards the solution, we may as well try to extrapolate each step a bit using a factor ω:

$$\Phi^{n+1}_{i,j,k} = \Phi^{n}_{i,j,k} + \omega\,(\Phi''^{n+1}_{i,j,k} - \Phi^{n}_{i,j,k})$$
$$= (1-\omega)\,\Phi^{n}_{i,j,k} + \omega\,\Phi''^{n+1}_{i,j,k}$$

- Gauss-Seidel is used for computing the solution $\Phi''^{n+1}_{i,j,k}$
- We need to have $0 < \omega < 2$. Otherwise, small scale errors grow
  - $\omega < 1$: *under relaxation*, can make unstable method stable
  - $\omega > 1$: *over relaxation*, accelerates the solution with N convergence
- SOR convergence scales linearly with grid length (N iterations; Jacobi N^2!)
- Optimal value of ω depends on grid resolution (Young 1950 and Frankel 1954)

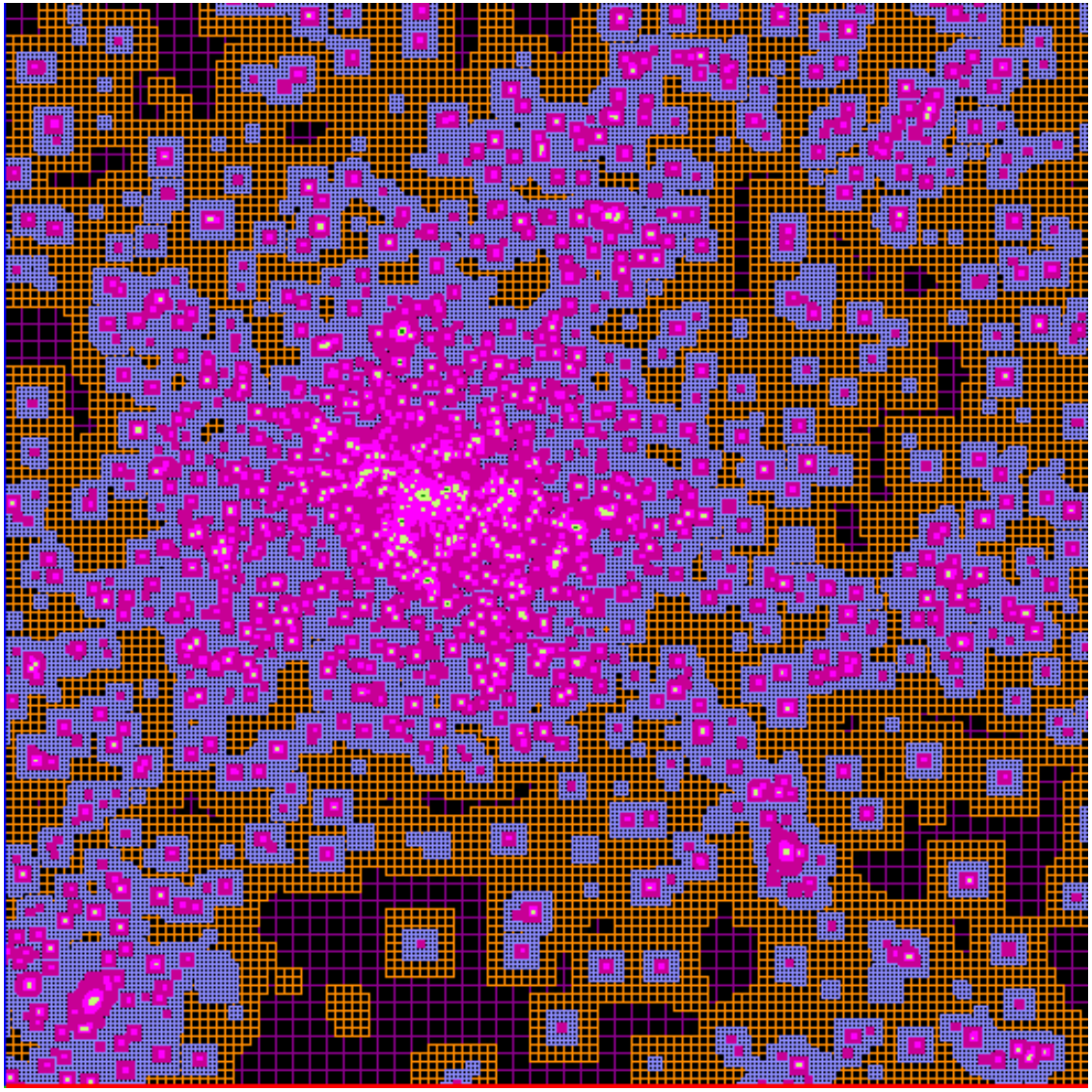
$$\omega = \frac{2}{1 + \sin\left(\frac{\pi}{N}\right)}$$

Summary Part I

- Self gravity plays a crucial role in many astrophysical applications; from the evolution of cosmological structure to planet formation
- Many methods exists to solve the Poisson equation, both direct and iterative. ***Cost scales super-linearly with resolution!***
- Direct methods are good for particle representations, but do not scale.
- The Fast Fourier Transform works well for uniform systems with very simple boundary conditions
- To solve the Poisson equation iterative methods are the most commonly used. They are attractive because there are no assumptions about boundary values.
- Jacobi, Gauss-Seidel, and SOR methods are examples of simple iterative methods, with SOR being the method of choice. ***SOR works well with up to ~128^3 meshes***

HST: 30 Doradus

Multi-scale problems have deep AMR grids



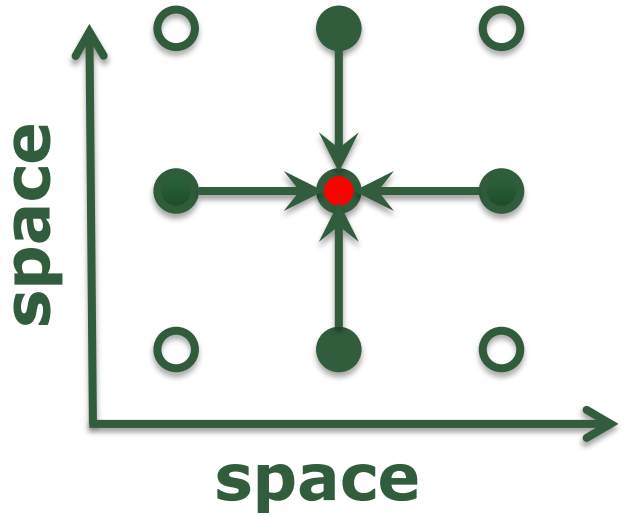
AMR grid from galaxy formation simulation (from RAMSES user-guide)

Iterative methods

- ***Jacobi Iterations:*** Take advantage of the linear nature of the equation:

1. Make a guess for the potential Φ
2. Assume this is the right solution, update iteratively as average of neighbors

$$6\,\Phi^{n+1}_{i,j,k} = \Phi^{n}_{i-1,j,k}+\Phi^{n}_{i+1,j,k} + \Phi^{n}_{i,j-1,k}+\Phi^{n}_{i,j+1,k} + \Phi^{n}_{i,j,k-1}+\Phi^{n}_{i,j,k+1} + - 4\pi G\rho\,\Delta x^2$$



- Problem is it only updates the shortest (grid-scale) wavelengths. Therefore the method is extremely sensitive to initial guess for the large-scale modes, and converges very slowly – iterations scale like N^2
- ***BUT!*** This does not scale to large grids, or adaptive resolution

Can we use iterative methods across grids?

- Use a simple method (Jacobi!) to solve for gravity inside a grid
- Update boundary conditions across grids
- In principle this does not scale either; only if we have the long wavelength modes of gravity right
- Remember: Problem is it only updates the shortest (grid-scale) wavelengths. Therefore, the method is extremely sensitive to initial guess for the large-scale modes, and converges very slowly – it takes $\mathcal{O}[(\#\text{grid points})^2]$ iterations
- This is the core problem addressed by Multi-grid methods

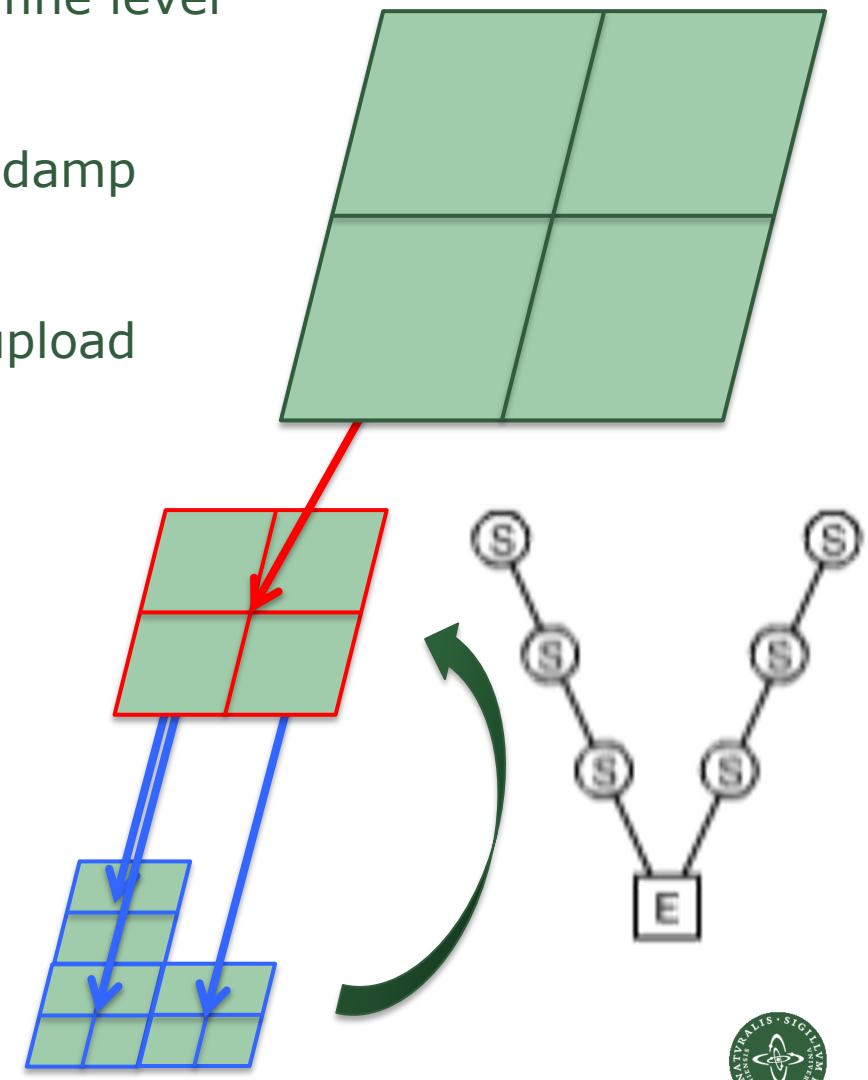
Multi-grid Method in a slide (two-way solve)

1. Use Jacobi or Gauss-Seidel cycle on fine level to damp high-frequency error

2. Restrict solution to coarse level and damp again

3. Use prolongation (interpolation) to upload *correction* from coarse to fine level

❑ Rinse and repeat!



Multi-grid Method: residual and error

Reformulation of the problem as linear algebra simplifies notation:

- The Poisson equation is like a linear equation with an extremely sparse matrix:

$$\Delta\Phi = 4\pi G\rho \iff A\mathrm{u} = S$$

where we have written $\Phi \equiv u$, $\Delta \equiv \mathrm{A}$ (a matrix representing the discrete Laplace operator, and $4\pi G\rho = S$.

- Assume that ***u*** is a solution to the problem, and ***v*** is the approximation obtained with e.g., a single Jacobi iteration. Then we can define

Error: $e = u - v$

Residual: $r = \mathrm{S} - Av$

- The key observation is that even if ***e*** is unknown, we can write an equation for it

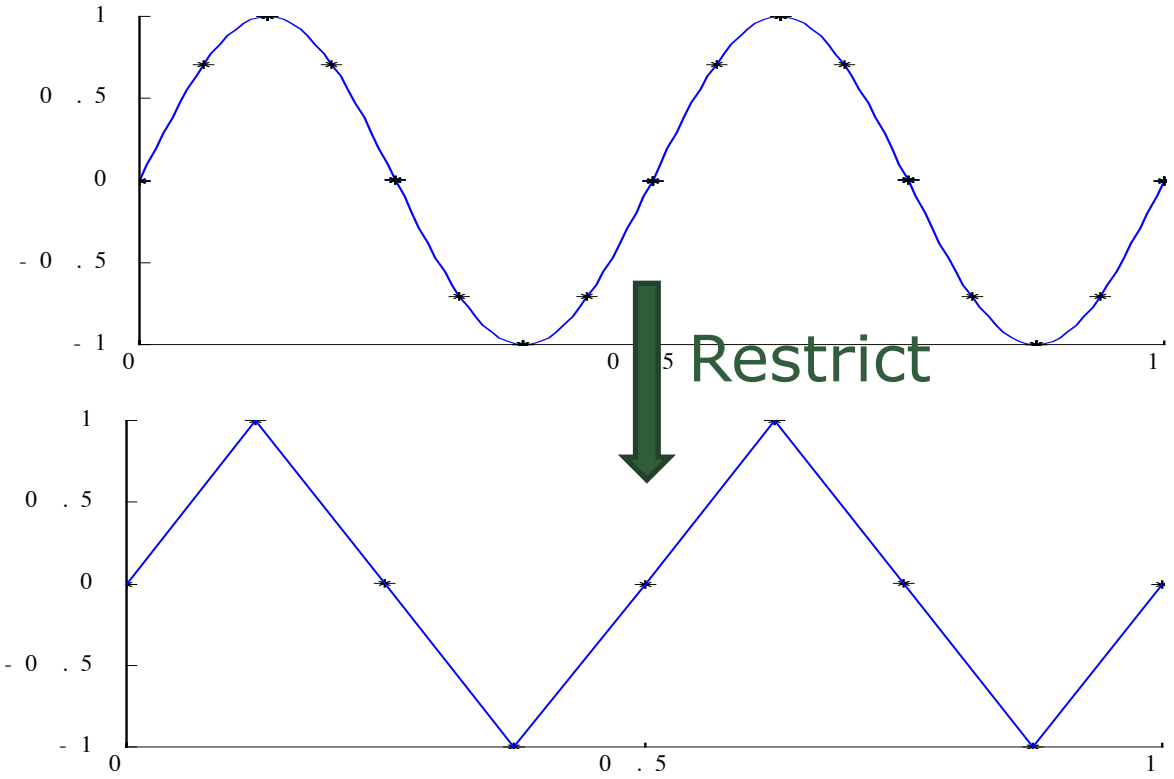
$$\mathrm{A}e = \mathrm{A}(u - v) = S - Av = r$$

- We have a new (but analogous) equation for the error term as the solution to the Poisson equation with ***r*** as a source!

Multi-grid Method: restriction - R

- If we solve for the error term, ***e***, we can use it to correct the solution, ***v***, and get closer to the real solution, ***u***.

- The iterative methods (Jacobi) correct the error at the high frequencies, but by restricting the solution to a coarse grained version of the equation, we can correct the error on all scales:

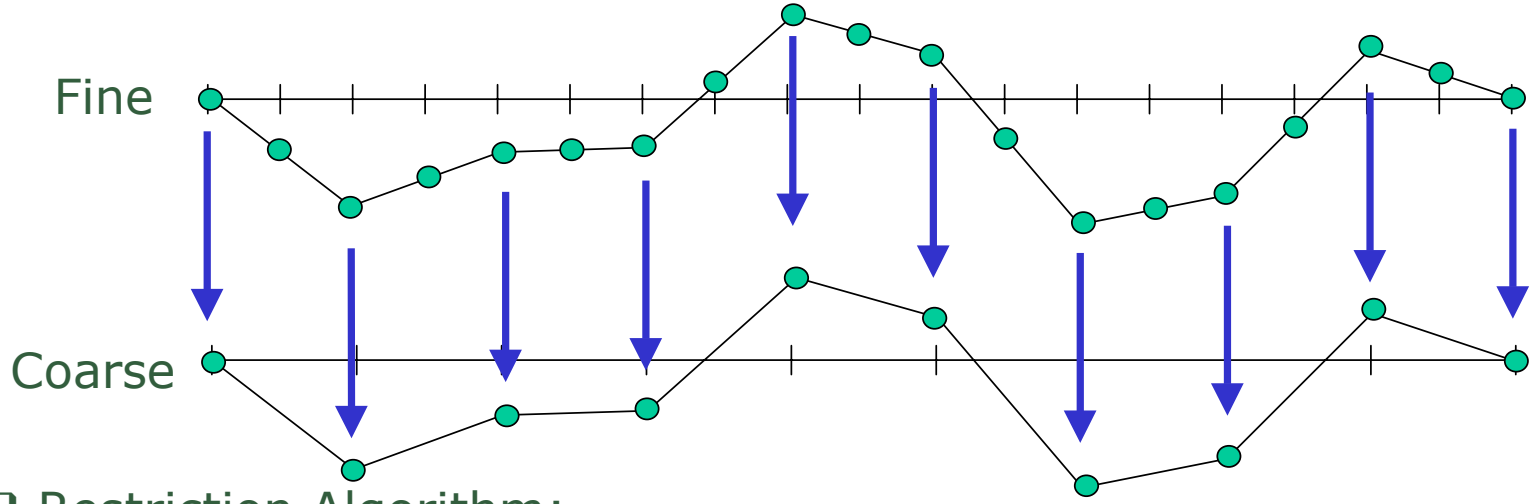


- Smooth features become sharper on a coarser mesh
- Low-frequency part of solution done through recursion
- Coarser meshes are also much cheaper to solve!

Multi-grid Method: restriction - R

- If we solve for the error term, ***e***, we can use it to correct the solution, ***v***, and get closer to the real solution, ***u***.

- There are several ways to restrict a mesh. The most straightforward is to simply sample every 2nd point, thereby killing all the highest Fourier modes, but not removing amplitude



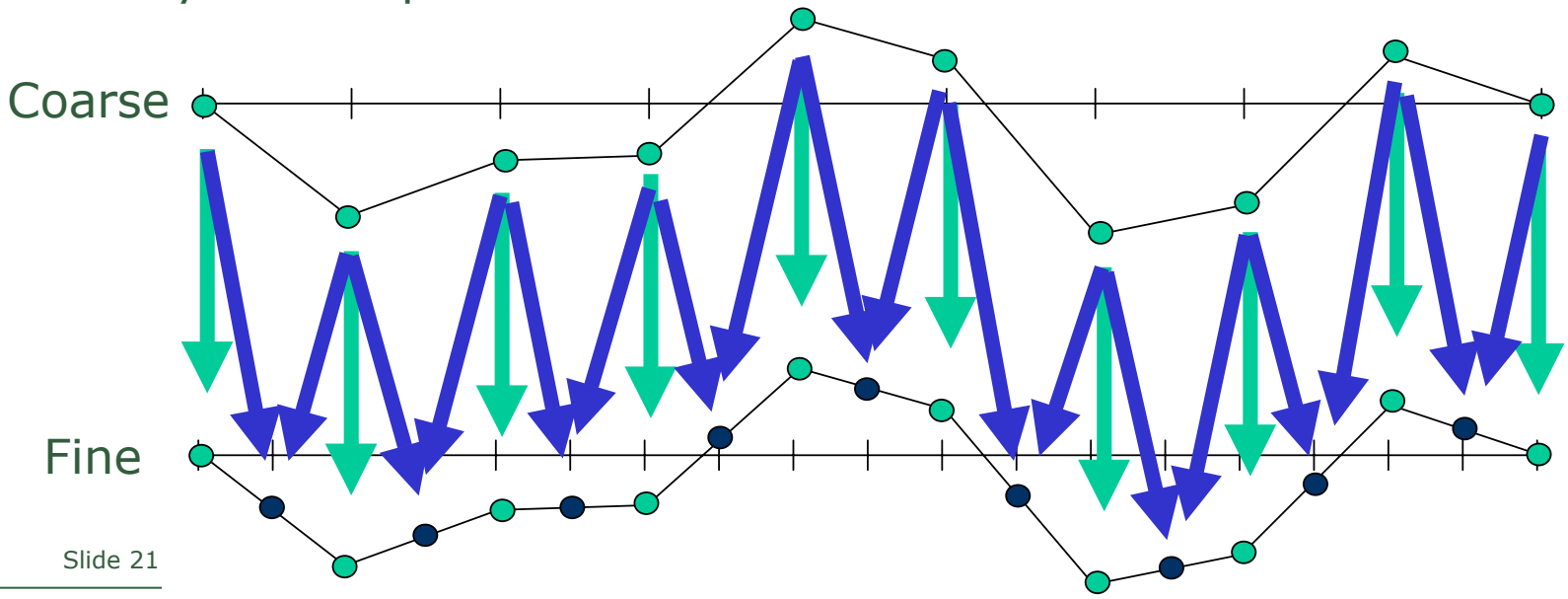
- Restriction Algorithm:
  - First solve on the fine grid. Calculate the residual r_0
  - Restrict r_0 to coarser grid $r_0 \rightarrow r_1$. Use r_1 as source to calculate the error on the coarse level with $A\, e_1 = r_1$

Multi-grid Method: prolongation - I

- If we solve for the error term, ***e***, we can use it to correct the solution, ***v***, and get closer to the real solution, ***u***.

- We need to get the error term on the coarse level back to the fine level. This is called *prolongation*. Essentially, we need to interpolate from coarse to fine level.

- Because we used straight injection, the simplest prolongation operator is to copy back every second point, and use average for every second point



Multi-grid Method: Fine – Coarse / R – I cycle

- If we solve for the error term, ***e***, we can use it to correct the solution, ***v***, and get closer to the real solution, ***u***.

- We now have a pseudo code for a full fine-coarse cycle:

1. Make an initial guess for solution on all levels: $\boldsymbol{v=0}$
2. Find solution on fine level: *Solve*($\boldsymbol{A\,u = S}$) → solution $\boldsymbol{v_0}$, residual $\boldsymbol{r_0}$
3. Restrict residual to coarse level: *R*($\boldsymbol{r_0}$) -> coarse level residual $\boldsymbol{r_1}$
4. Find solution for error on coarse level: *Solve*($\boldsymbol{A\,e = r_1}$) → $\boldsymbol{v_1}$, $\boldsymbol{r_1}$
5. Prolongate error to fine level: *I*($\boldsymbol{v_1}$) → fine level error $\boldsymbol{e_0}$
6. Correct solution on fine level: $\boldsymbol{v_0}$ -> $\boldsymbol{v_0 + e_0}$
7. Solve on fine level again with new $\boldsymbol{v_0}$
8. Check residual (compared to ***S***). If not good enough, go back to 2.

Multi-grid Method: Fine – Coarse / R – I cycle

- If we solve for the error term, ***e***, we can use it to correct the solution, ***v***, and get closer to the real solution, ***u***.

- Observations:
  - Restriction works best if solution is smooth
    → Important to first remove high frequency noise with a few Jacobi iterations (typically 1 or 2 is enough)

  - When back at fine level after a fine-coarse iteration a few more iterations will kill even more of high-frequency noise
    → have to iterate multi-grid cycle several times to reach convergence

  - This can be extended to many levels with restriction all the way to 2x2x2 grids. E.g.: (R: Restrict, P: Prolongate, number: level)
    $(\text{Solve, R})_0 \rightarrow (\text{Solve, R})_1 \rightarrow \ldots \rightarrow (\text{Solve, R})_{L-1} \rightarrow (\text{Solve})_L \rightarrow$
    $(\text{P})_L \rightarrow (\text{Solve,P})_{L-1} \rightarrow \ldots \rightarrow (\text{Solve,P})_1 \rightarrow (\text{Solve})_0$

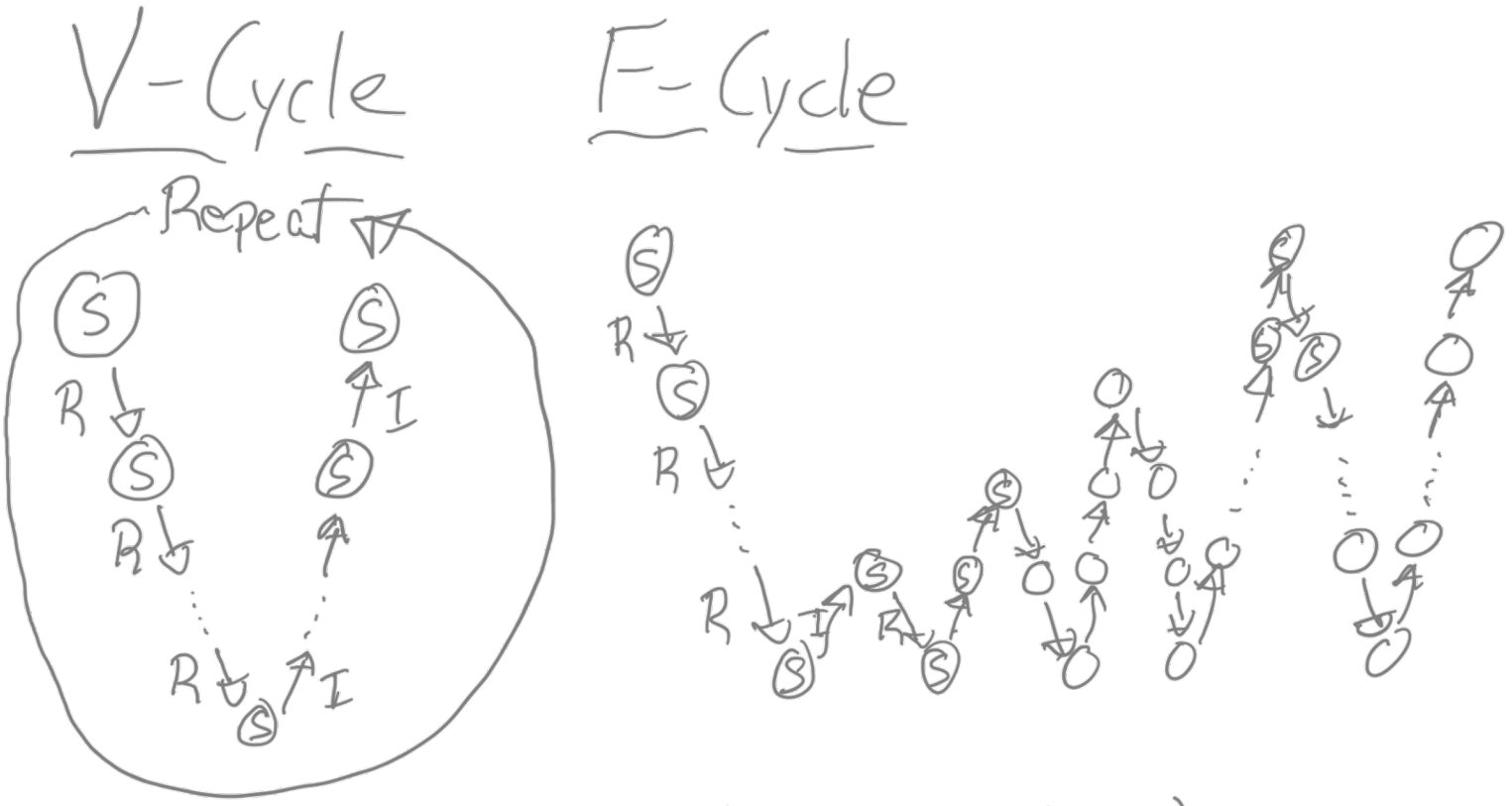
Full Multigrid Method

- The complete algorithm in pseudo code for a full multigrid cycle:

1. Make an initial guess for solution on all levels: $\boldsymbol{v=0}$
2. Find solution on fine level: *Solve*($\boldsymbol{A\,u = S}$) → solution $\boldsymbol{v_0}$, residual $\boldsymbol{r_0}$
3. Restrict residual to coarse level: $R(\boldsymbol{r_0})$ -> coarse level residual $\boldsymbol{r_1}$
4. Find solution for error on coarse level: *Solve*($\boldsymbol{A\,e = r_1}$) → $\boldsymbol{v_1}$, $\boldsymbol{r_1}$
5. …Repeat steps 3+4 on increasingly coarser levels
6. Prolongate error from a coarse to a finer level: $I(\boldsymbol{v_L})$ → fine level error $\boldsymbol{e_{L-1}}$
7. Correct solution on finer level: $\boldsymbol{v_{L-1}}$ -> $\boldsymbol{v_{L-1} + e_{L-1}}$
8. Solve on fine level again with new $\boldsymbol{v_{L-1}}$
9. …Repeat steps 6+7+8 until reaching the finest level (level 0)
10. Check residual (compared to $\boldsymbol{S}$). If not good enough, go back to 2, but now we have guesses for $\boldsymbol{v}$ ($\boldsymbol{v_0, v_1, \ldots v_L}$) on all $\boldsymbol{L}$ levels. Residuals and error terms will be smaller.

This recursion towards first increasingly coarser levels and then increasingly finer levels can be seen as a V-cycle.

Different Multigrid solution strategies



S: smooth (Solve $Ae=r$ to get v,r)
R: restrict residual, r, to coarse resolution
I: prolongate solution, v, to fine level

more exists, f.x. the W-cycle

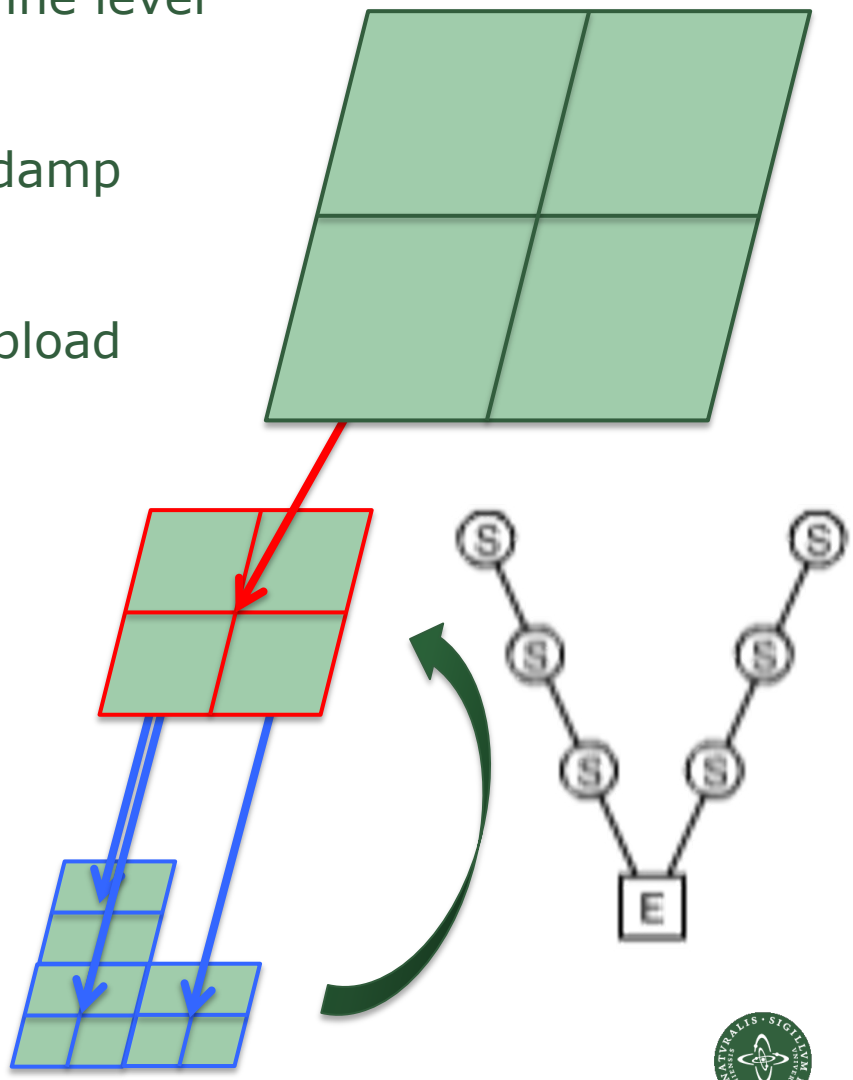
Multi-grid Method in a slide (two-way solve)

1. Use Jacobi or Gauss-Seidel cycle on fine level to damp high-frequency error

2. Restrict solution to coarse level and damp again

3. Use prolongation (interpolation) to upload *correction* from coarse to fine level

- In principle converges in O(N) time
- Works without too much trouble on AMR meshes and with arbitrary boundary conditions
- But large communication overhead if boundary is complex
- Method of choice in many AMR codes



Summary Part II

- Self gravity plays a crucial role in many astrophysical applications; from cosmological structure formation to planet formation
- Self gravity in cold systems naturally makes problems multi-scale to the extreme
- Many methods exists to solve the Poisson equation, both direct and iterative
- The Fast Fourier Transform is attractive for uniform systems and can be used at the coarsest level in AMR
- The multi-grid method is dominantly used with AMR codes
- See: https://www.caam.rice.edu/~caam551/mgtut.pdf for a very thorough (+100 slides!) walk-through by one of the "fathers of multi-grid methods" – a few of the illustrations came from there.