FLASH Code Tutorial

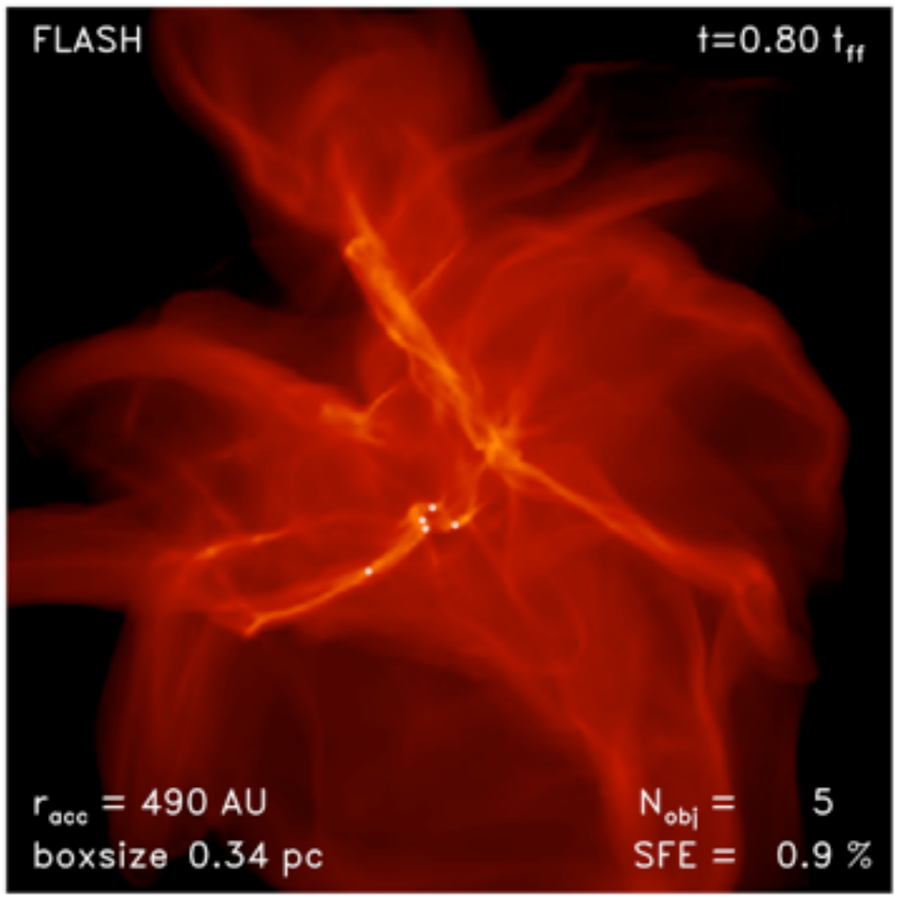
part II
Gravity modules

Robi Banerjee
Hamburger Sternwarte
banerjee@hs.uni-hamburg.de

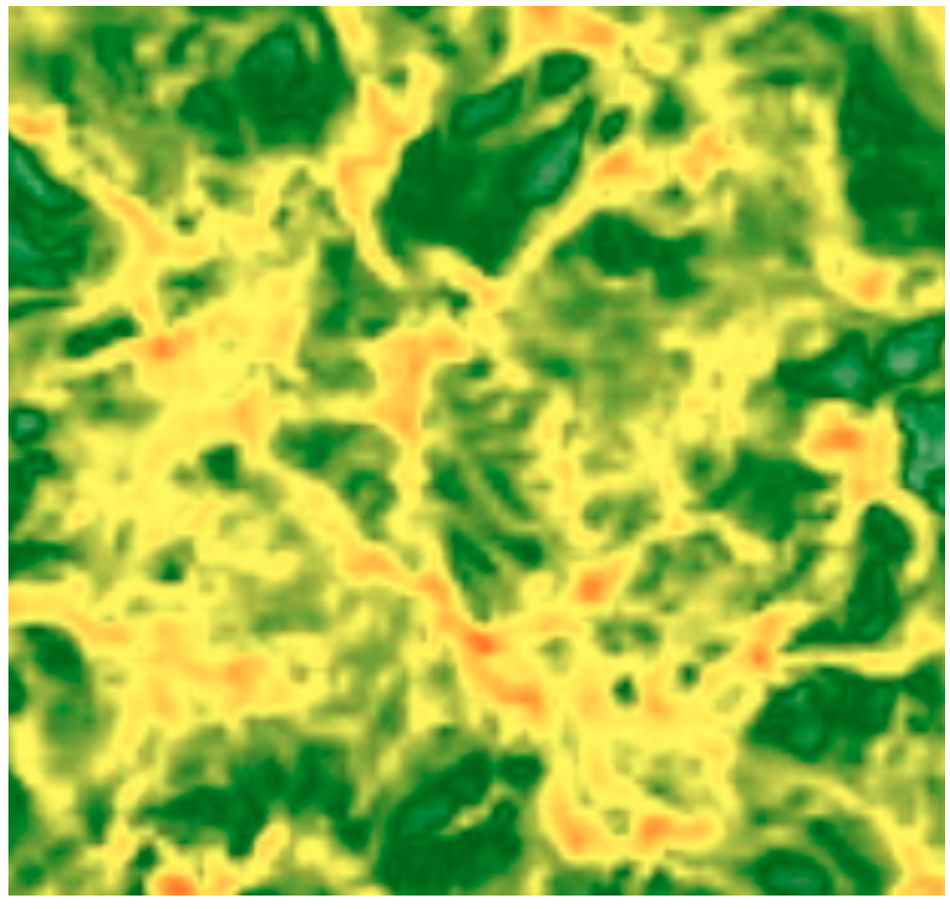
FLASH code: Gravity

Motivation

- dynamics of stratified atmospheres
- N-body dynamics in gaseous media (e.g. stellar cluster, planets)
- collapse of gas cores (e.g. star formation)
- formation of molecular clouds
- ...



collapse of a cloud core



molecular cloud formation

FLASH code: Gravity

- coupling to hydro-dynamics / MHD

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) + \nabla P = \rho \mathbf{g}$$

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

$\mathbf{g}$: gravitational acceleration

$\Longrightarrow$ via source term

$\Longrightarrow$ no explicit total energy conservation using gravitational potential

FLASH code: Gravity

- Gravity modules
  - `source/physics/Gravity`

    $\Longrightarrow$ default: constant gravitational acceleration

  - `source/Grid/GridSolvers`

    $\Longrightarrow$ for self-gravity

    - `Multipole`
    - `Multigrid`
    - `BHTree`

FLASH code: Gravity

- time-independent external fields:
  - source/`physics/Gravity/GravityMain`
    - `Constant`

      $\Longrightarrow \quad \mathbf{g} = \text{const}$ in `gdirec` $= x,y$ or z direction

    - `PlanPar`

      $\Longrightarrow \quad \mathbf{g} = G_N M / h^2$ h = x,y or z-direction

    - `PointMass`

      $\Longrightarrow \quad \mathbf{g} = G_N M / r^2 \, \mathbf{r}/r$

      $\Longrightarrow$ similarly for *sink particles*

FLASH code: Gravity

- Self-gravity:

$\Longrightarrow$ `source/Grid/GridSolvers`

provides gravitational potential $\Phi(\mathbf{x})$ via

- multi-pole solver
- multi-grid solver (Paul Ricker 2008)
- tree-based solver (Richard Wunsch)

$$\Longrightarrow \quad \nabla^2 \Phi(\mathbf{x}) = 4\pi G_N \rho(\mathbf{x}) \qquad \Longrightarrow \quad \mathbf{g}(\mathbf{x}) = -\nabla \Phi(\mathbf{x})$$

$$\Longrightarrow \quad \begin{aligned} g_{x;ijk} &= \frac{1}{2\Delta x}\left(\phi_{i-1,j,k} - \phi_{i+1,j,k}\right) \\ g_{y;ijk} &= \frac{1}{2\Delta y}\left(\phi_{i,j-1,k} - \phi_{i,j+1,k}\right) \\ g_{z;ijk} &= \frac{1}{2\Delta z}\left(\phi_{i,j,k-1} - \phi_{i,j,k+1}\right) \end{aligned}$$

implementation via finite difference scheme

FLASH code: Gravity

- setup with self-gravity:

  in `Config` file

  ⟹ `REQUIRES physics/Gravity/GravityMain/Poisson/Multipole`

  or

  ⟹ `REQUIRES physics/Gravity/GravityMain/Poisson/Multigrid`

  or

  ⟹ `REQUIRES physics/Gravity/GravityMain/Poisson/BHTree`

FLASH code: Gravity

- setup with self-gravity:

  boundary conditions

bcTypes	Type of boundary condition
0	Isolated boundaries
1	Periodic boundaries
2	Dirichlet boundaries
3	Neumann boundaries
4	Given-value boundaries

FLASH code: Gravity

- Multipole solver

formal solution of $\nabla^2\Phi(\mathbf{x}) = \alpha\,\rho(\mathbf{x})$; $\alpha = 4\pi G_N$

$$\Longrightarrow \qquad \phi(\mathbf{x}) = -\frac{\alpha}{4\pi}\int d^3\mathbf{x}'\,\frac{\rho(\mathbf{x}')}{|\mathbf{x}-\mathbf{x}'|}$$

with Green's function

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

where $Y_{lm}(\theta,\varphi)$ are the spherical harmonic functions

$\Longrightarrow$ origin at center of mass (CM): $\mathbf{x}_{\rm cm} = \dfrac{\int \mathbf{x}\rho(\mathbf{x})\,d\mathbf{x}}{\int \rho(\mathbf{x})\,d\mathbf{x}}$

FLASH code: Gravity

- Multipole solver

$$\Longrightarrow \quad \Phi(\mathbf{x}) \;\rightarrow\; \Phi(r, \mu(r))$$

with source moments $\mu(r)$

e.g. $$\mu^{\rm ei}_{\ell m}(r) \;\equiv\; \frac{(\ell-m)!}{(\ell+m)!}\int_{r>r'} d^3\mathbf{x}'\, r'^{\ell}\, \rho(\mathbf{x}')\, P_{\ell m}(\cos\theta')\cos m\varphi'$$

and $\mu^{\rm oi}, \mu^{\rm eo}, \mu^{\rm oo}$, (even/odd, inner/outer)

$\Longrightarrow$ compute μ's up to $l_{\rm max}$

$\Longrightarrow$ **note**: scales at least as $2\times N_{\rm grid}\times {l_{\rm max}}^2$

$\Longrightarrow$ only useful for low $l_{\rm max}$, i.e. nearly spherical problems

FLASH code: Gravity

- Multipole solver
  - main runtime parameters

Variable	Type	Default	Description
`mpole_lmax`	integer	10	Maximum multipole moment
`quadrant`	logical	`.false.`	Use symmetry to solve a single quadrant in 2D axisymmetric cylindrical (r, z) coordinates, instead of a half domain.

- supported grid geometries:

  $\Longrightarrow$ 1D, spherical
  $\Longrightarrow$ 2D, cylindrical, spherical
  $\Longrightarrow$ 3D, Cartesian, axi-symmetric

FLASH code: Gravity

- Multipole solver
  - possible test suit:

    `unitTest/Multipole`

    $\Longrightarrow$ only multipole solver without gravity
    based on MacLaurin spheroid analytic solution

FLASH code: Gravity

- Multipole solver
  - improved Multipole solver

    $\Longrightarrow$ `source/Grid/GridSolvers/Multipole_new`

  - improvements:
    - efficient memory layout
    - elimination of over- and underflow errors when using astrophysical (dimensions $\approx 10^9$) domains
    - elimination of subroutine call overhead (1 call per cell)
    - ...

FLASH code: Gravity

- Multigrid solver

  - based on improved *Huang-Greengard* (2000) method by *P. Ricker* (ApJS, 2008)

  - iterative method
  - uses fine-coarse structure of AMR grid

  $\Longrightarrow$ restrict & interpolate solution
  to different refinement levels $\Longrightarrow$ **V-cycle**

- main functions in

  `source/Grid/GridSolvers/Multigrid/`
  $\Longrightarrow$ `gr_hg*.F90`

FLASH code: Gravity

- Multigrid solver: V-cycle

```
!! DESCRIPTION
!!  This is the main Poisson solve routine for the Huang & Greengard
!!  (2000, SIAM J. Sci. Comput., 21, 1551) algorithm.  This routine
!!  defines the multigrid cycle as expressed in the article.
!!
!! coarse ^
!!        o---->o                    o---->o
!!         ^    s \                  ^    s \ c
!!       r|     o o                 |r    o o o
!!       e|      l \                |e     l \ r
!!       s|       v o               |s      v o r
!!       t|        e \              |t       e \ e
!!       r|           o             |r          o c
!!       i|            \            |i           \ t
!!       c|             o           |c            o
!!       t|              \          |t             \
!!  src  |               o----->o gr_iSls          o------>...until |r| < rtol
!! fine  |_____________take residual_____________take new residual__>
!!
```

FLASH code: Gravity

- Multigrid solver: more details
  - residual $$R(\mathbf{x}) \equiv 4\pi G\rho(\mathbf{x}) - \nabla^2\tilde{\phi}(\mathbf{x})$$
  - finite difference operator:

$$\mathcal{D}_\ell\tilde{\phi}^{b\ell}_{ijk} \equiv \frac{1}{\Delta x_\ell^2}\left(\tilde{\phi}^{b\ell}_{i+1,jk} - 2\tilde{\phi}^{b\ell}_{ijk} + \tilde{\phi}^{b\ell}_{i-1,jk}\right) + \ldots$$

- restrict operator: fine → coarse

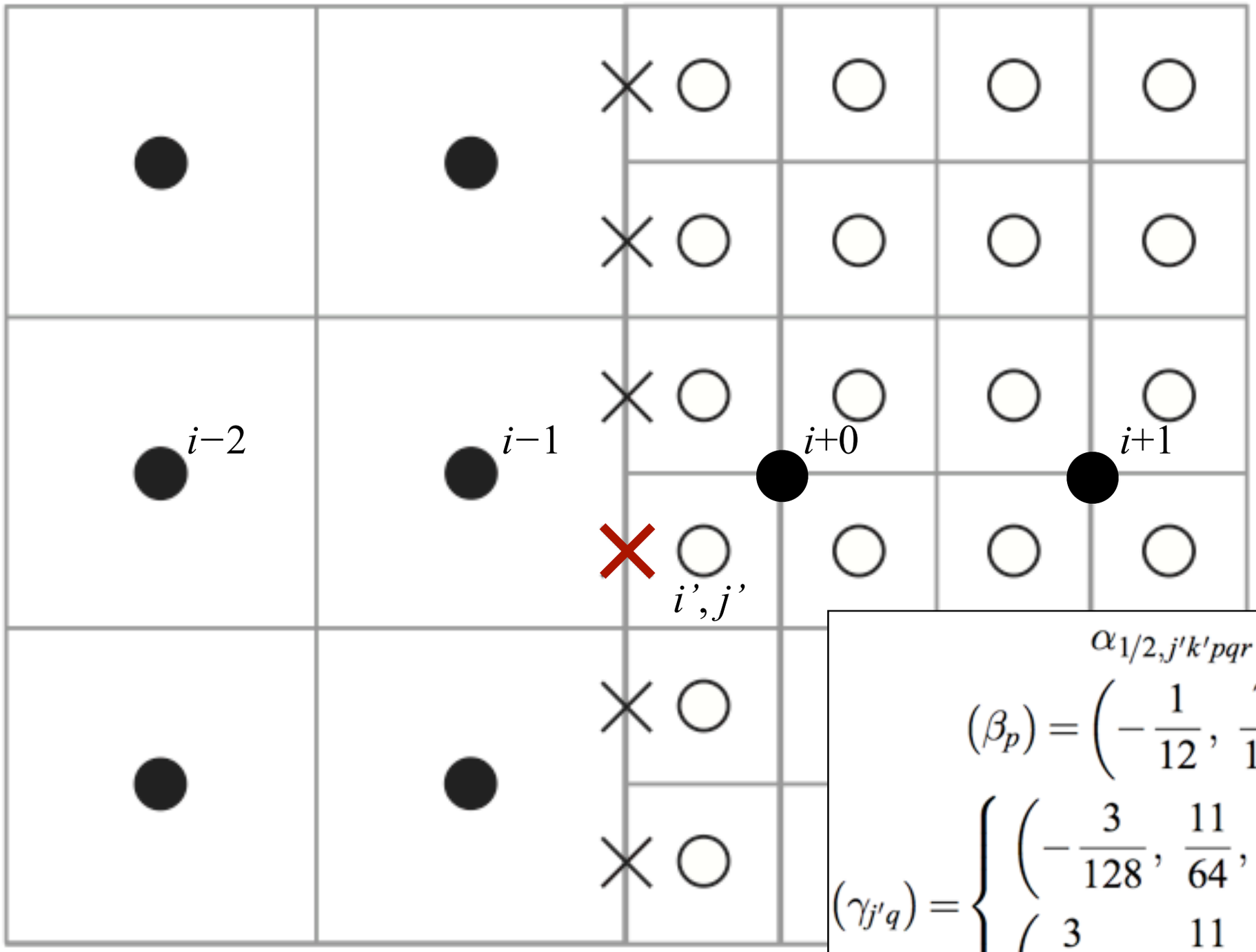
$$(\mathcal{R}_\ell\tilde{\phi})^{\mathcal{P}(c),\ell}_{ijk} \equiv \frac{1}{2^d}\sum_{i'j'k'}\tilde{\phi}^{c,\ell+1}_{i'j'k'}$$

- interpolation operator at block edges: coarse → fine

$$(\mathcal{I}_\ell\tilde{\phi})^{c,\ell+1}_{i'j'k'} \equiv \sum_{p,q,r=-2}^{2}\alpha_{i'j'k'pqr}\tilde{\phi}^{\mathcal{P}(c),\ell}_{i+p,j+q,k+r}$$

FLASH code: Gravity

- Multigrid solver: Interpolation operator



Example:
$-x$ boundary

$$\alpha_{1/2,j'k'pqr} = \beta_p \gamma_{j'q} \gamma_{k'r},$$

$$(\beta_p) = \left(-\frac{1}{12}, \frac{7}{12}, \frac{7}{12}, -\frac{1}{12}, 0\right),$$

$$(\gamma_{j'q}) = \begin{cases} \left(-\frac{3}{128}, \frac{11}{64}, 1, -\frac{11}{64}, \frac{3}{128}\right), & j' \text{ odd}, \\ \left(\frac{3}{128}, -\frac{11}{64}, 1, \frac{11}{64}, -\frac{3}{128}\right), & j' \text{ even} \end{cases}$$

FLASH code: Gravity

- Multigrid solver: more details

1. Restrict the source function $4\pi G\rho$ to all levels. Subtract the global average for the `periodic` case.

2. *Interpolation step:* For ℓ from 1 to $\ell_{\max}$,

   (a) Reset the grid so that ℓ is the maximum refinement level

   (b) Solve $\mathcal{D}_\ell \tilde{\phi}^{b\ell}_{ijk} = 4\pi G\rho^{b\ell}_{ijk}$ for all blocks b on level ℓ. → sin-type Green's function or FFT

   (c) Compute the residual $R^{b\ell}_{ijk} = 4\pi G\rho^{b\ell}_{ijk} - \mathcal{D}_\ell \tilde{\phi}^{b\ell}_{ijk}$

   (d) For each block b on level ℓ that has children, prolong face values for $\tilde{\phi}^{b\ell}_{ijk}$ onto each child block.

3. *Residual propagation step:* Restrict the residual $R^{b\ell}_{ijk}$ to all levels.

4. *Correction step:* Compute the discrete L_2 norm of the residual over all leaf-node blocks and divide it by the discrete L_2 norm of the source over the same blocks. If the result is greater than a preset threshold value, proceed with a correction step: for each level ℓ from 1 to $\ell_{\max}$,

   (a) Reset the grid so that ℓ is the maximum refinement level

   (b) Solve $\mathcal{D}_\ell C^{b\ell}_{ijk} = R^{b\ell}_{ijk}$ for all blocks b on level ℓ. → sin-type Green's function or FFT

   (c) Overwrite $R^{b\ell}_{ijk}$ with the new residual $R^{b\ell}_{ijk} - \mathcal{D}_\ell C^{b\ell}_{ijk}$ for all blocks b on level ℓ.

   (d) Correct the solution on all leaf-node blocks b on level ℓ: $\tilde{\phi}^{b\ell}_{ijk} \to \tilde{\phi}^{b\ell}_{ijk} + C^{b\ell}_{ijk}$.

   (e) For each block b on level ℓ that has children, interpolate face boundary values of $C^{b\ell}_{ijk}$ for each child.

5. If a correction step was performed, return to the residual propagation step.

FLASH code: Gravity

- Multigrid solver: more details

1. Restrict the source function $4\pi G\rho$ to all levels. Subtract the global average for the `periodic` case.

2. *Interpolation step:* For ℓ from 1 to $\ell_{\max}$,

   (a) Reset the grid so that ℓ is the maximum refinement level

   (b) Solve $\mathcal{D}_\ell \tilde{\phi}^{b\ell}_{ijk} = 4\pi G\rho^{b\ell}_{ijk}$ for all blocks b on level ℓ. → sin-type Green's function or FFT

   (c) Compute the residual $R^{b\ell}_{ijk} = 4\pi G\rho^{b\ell}_{ijk} - \mathcal{D}_\ell \tilde{\phi}^{b\ell}_{ijk}$

   (d) For each block b on level ℓ that has children, prolong face values for $\tilde{\phi}^{b\ell}_{ijk}$ onto each child block.

3. *Residual propagation step:* Restrict the residual $R^{b\ell}_{ijk}$ to all levels.

4. *Correction step:* Compute the discrete L_2 norm of the residual over all leaf-node blocks and divide it by the discrete L_2 norm of the source over the same blocks. If the result is greater than a preset threshold value, proceed with a correction step: for each level ℓ from 1 to $\ell_{\max}$,

   (a) Reset the grid so that ℓ is the maximum refinement level

   (b) Solve $\mathcal{D}_\ell C^{b\ell}_{ijk} = R^{b\ell}_{ijk}$ for all blocks b on level ℓ. → sin-type Green's function or FFT

   (c) Overwrite $R^{b\ell}_{ijk}$ with the new residual $R^{b\ell}_{ijk} - \mathcal{D}_\ell C^{b\ell}_{ijk}$ for all blocks b on level ℓ.

   (d) Correct the solution on all leaf-node blocks b on level ℓ: $\tilde{\phi}^{b\ell}_{ijk} \to \tilde{\phi}^{b\ell}_{ijk} + C^{b\ell}_{ijk}$.

   (e) For each block b on level ℓ that has children, interpolate face boundary values of $C^{b\ell}_{ijk}$ for each child.

5. If a correction step was performed, return to the residual propagation step.

FLASH code: Gravity

```
do m = 1, gr_hgMeshRefineMax
   call gr_hgSetMaxLevel(m)    ! This is a new FLASH3 routine
   ! call gr_hgNorm(0, 1, gr_iSource, norm_tmp, MG_NODES_LEAF_ONLY)
   call gr_hgSetZeroBoundary(m, gr_iSoln)  !oK
   call gr_hgSolveLevel(m, gr_iSource, gr_iSoln, SolveBlock, MG_NODES_ALL_NODES)  ! Working here.....
   call gr_hgResidual(m, gr_iSource, gr_iSoln, gr_iSls) !oK
   call gr_hgProlongBndries(m, gr_iSoln, gr_iSoln, 0)  ! LBR doesn't wanna know.  But we looked at it...
enddo
! Correction step.  Restrict residuals from finer levels to coarser levels.
! Solve for correction on these levels and interpolate boundary conditions to
! finer levels.  Solve for corrections there and apply.  Repeat.  Repeat these
! correction steps until the desired residual norm is achieved.
do n = 0, gr_hgMaxCorrections
   call gr_hgNorm(0, 2, gr_iSls, norm_lhs, MG_NODES_LEAF_ONLY)

   if (norm_lhs/norm_rhs <= gr_hgMaxResidualNorm) exit

   do m = gr_hgMeshRefineMax-1, 1, -1
      call gr_hgRestrict(m+1, gr_iSls, gr_iSls)
   enddo

   call gr_hgSetMaxLevel(1)
   call gr_hgSetZeroBoundary(1, gr_iCorr)
   call gr_hgSolveLevel(1, gr_iSls, gr_iCorr, SolveBlock, MG_NODES_ALL_NODES, dt, chi)
   call gr_hgLevelAdd(1, gr_iSoln, gr_iCorr, MG_NODES_LEAF_ONLY) !oK
   call gr_hgProlongBndries(1, gr_iCorr, gr_iCorr, 0)  !infamous

   do m = 2, gr_hgMeshRefineMax
      call gr_hgSetMaxLevel(m)
      call gr_hgSetZeroBoundary(m, gr_iCorr)
      call gr_hgSolveLevel(m, gr_iSls, gr_iCorr, SolveBlock, MG_NODES_ALL_NODES)
      call gr_hgResidual(m, gr_iSls, gr_iCorr, gr_iSls)  ! not called on top level
      call gr_hgLevelAdd(m, gr_iSoln, gr_iCorr, MG_NODES_LEAF_ONLY)
      call gr_hgProlongBndries(m, gr_iCorr, gr_iCorr, 0)
   enddo

enddo
```

FLASH code: Gravity

- Multigrid solver: Additional fields needed:
  - ‘`gpot`’ key `GPOT_VAR`: actual gravitational potential
  - ‘`gpol`’ key `GPOT_VAR`: gravitational potential from previous time step $\Longrightarrow$ needed to start solution iteration
  - ‘`isls`’ key `ISLS_VAR`: residual potential variable
  - ‘`icor`’ key `ICOR_VAR`: correction potential variable
  - ‘`imgm`’ key `IMGM_VAR`: image mass for periodic BCs
  - ‘`imgp`’ key `IMGP_VAR`: image potential for periodic BCs

FLASH code: Gravity

- Multigrid solver:
  - supported geometries

    $\Longrightarrow$ only Cartesian geometry in 1, 2 and 3 D
  - supported boundary conditions

    $\Longrightarrow$ isolated

    $\Longrightarrow$ perodic

    $\Longrightarrow$ Dirichlet (Φ given at the boundaries)

    $\Longrightarrow$ Neumann, not yet implemented

FLASH code: Gravity

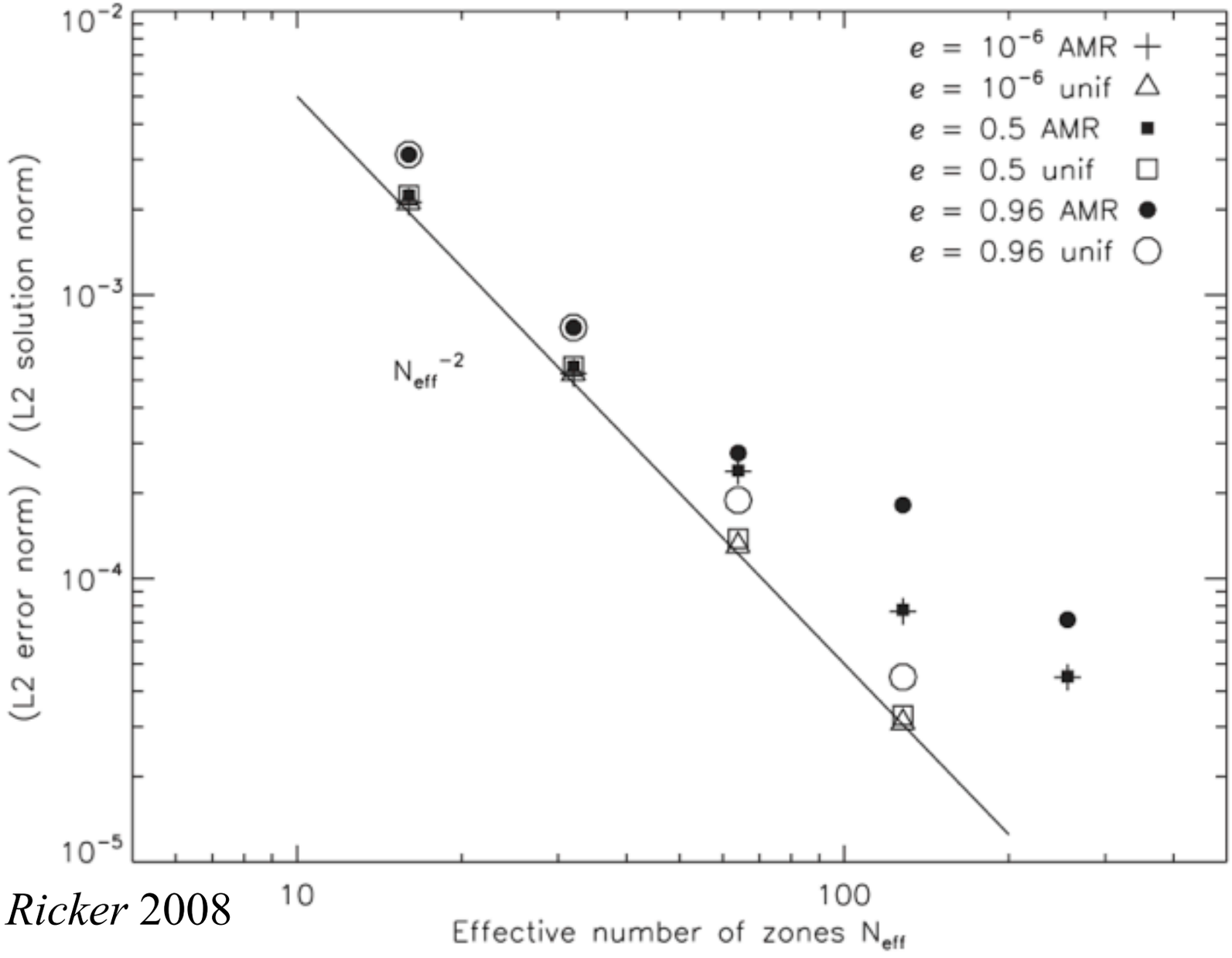
- Multigrid solver: main files

`source/Grid/GridSolvers/Multigrid`

```
gr_hgBndry.F90
gr_hgData.F90
gr_hgFinalize.F90
gr_hgGuardCell.F90
gr_hgInit.F90
gr_hgInitSource.F90
gr_hgLevelAdd.F90
gr_hgLevelAddScalar.F90
gr_hgLevelMultiplyScalar.F90
gr_hgLevelZero.F90
gr_hgMapBcType.F90
gr_hgNorm.F90
gr_hgPoissonSolve1D.F90
gr_hgPoissonSolve2D.F90
gr_hgPoissonSolve3D.F90
gr_hgPoissonSolveBlock.F90
gr_hgProlongBndries.F90
gr_hgRecordNodeTypeState.F90
gr_hgResidual.F90
gr_hgRestoreNodeTypes.F90
gr_hgRestrict.F90
gr_hgSetExtBoundary.F90
gr_hgSetMaxLevel.F90
gr_hgSetZeroBoundary.F90
gr_hgSolve.F90
gr_hgSolveLevel.F90
gr_hg_amr_1blk_bcset_work.F90
```

FLASH code: Gravity

- Multigrid solver: Solution convergence
  $\Longrightarrow$ UG: truncation error: $O(\Delta x^2)$
  $\Longrightarrow$ weaker for AMR (here: 5% of vol. on highest refinement level)



Ricker 2008

FLASH code: Gravity

- Barns & Hut Tree Solver
  (*Barns & Hut*, Nature 1986)

  - implemented by Richard Wünsch (Prague)
  - based on an oct-tree with monopole moments
    $\Longrightarrow$ matches oct-tree AMR structure of PARAMESH

- basic properties
  - scaling: $N \log(N)$: N number of grid/mass cells
    instead N^2 of for direct summation

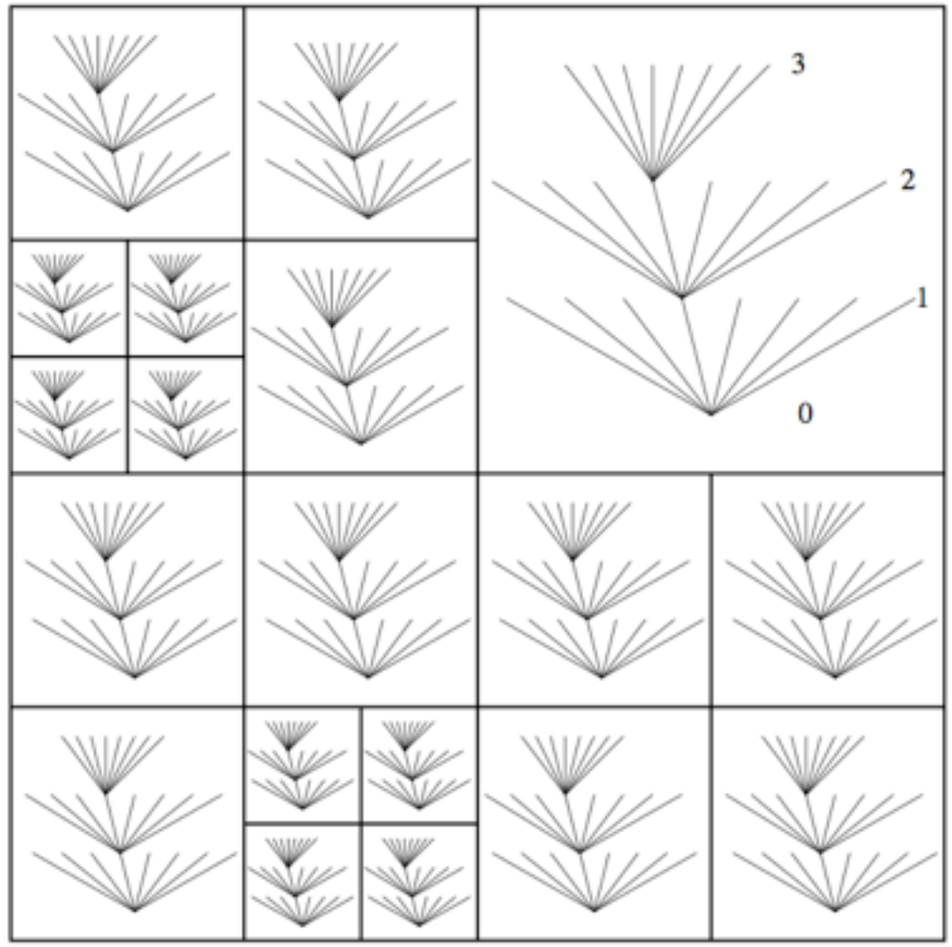
  - works only for `NBX = NBY = NBZ`
  - `NBX` $= 2^n$ (not yet tested for `NBX` $\neq 8$)

FLASH code: Gravity

- Tree Solver: steps to calculate the gravitational potential:
  - build the tree:
    1. build oct-tree in block (block-tree)
       $\Longrightarrow \log_2(\texttt{nbx})$ levels
       $\Longrightarrow$ stored in `gr_bhTreeArray`

    2. masses and CMs of top nodes are distributed to all processors via
       `gr_bhTreeParentTree`
       `(4,MAXBLOCKS,NPROCS)`

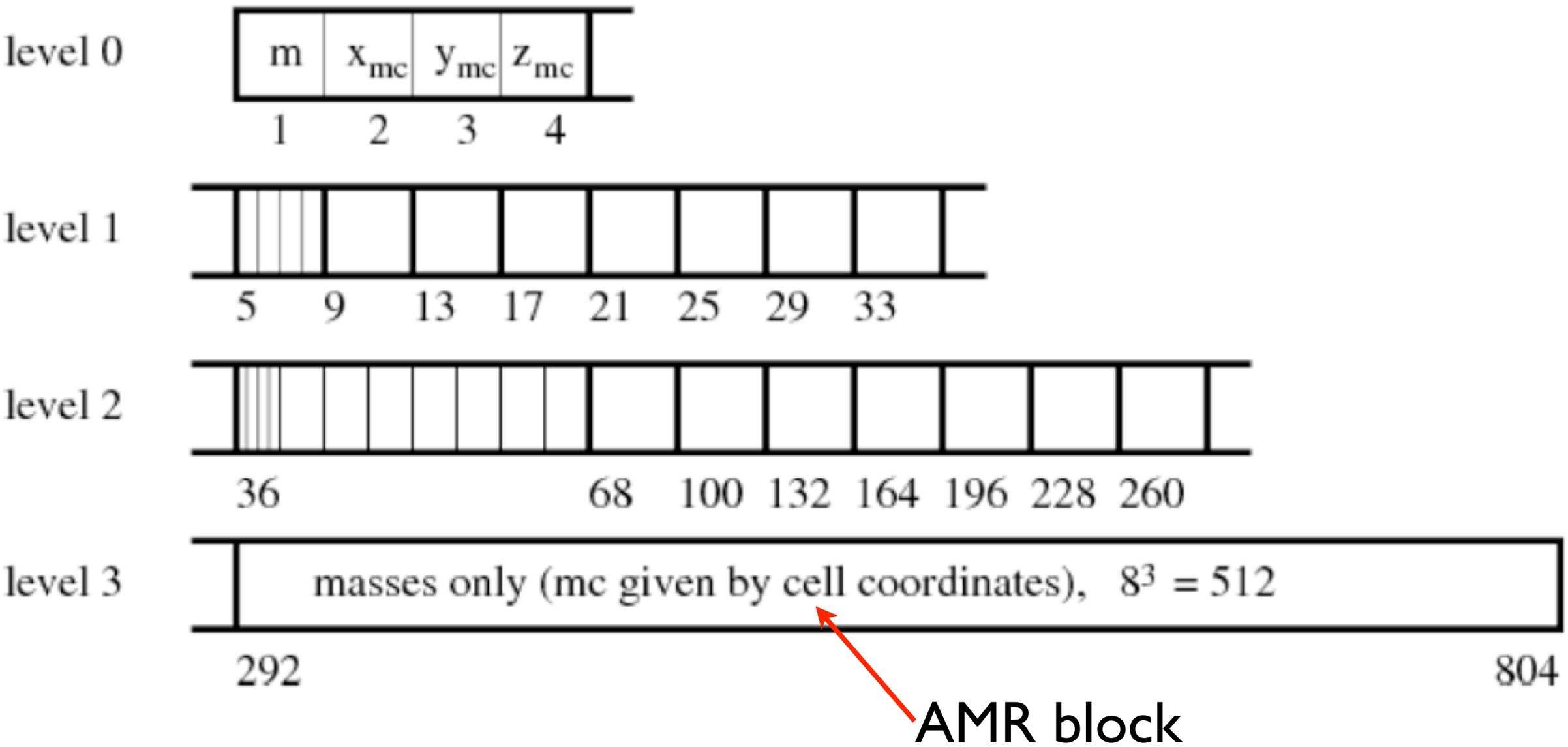
    3. calculation of masses and CMs of parent blocks on all CPUs
       $\Longrightarrow$ stored also in `gr_bhTreeParentTree`



$\Longrightarrow$ tree down to leaf block information on all CPUs

FLASH code: Gravity

- block-tree structure in memory for
  blocks with $8\times8\times8$ grid cells
  $\Longrightarrow$ linear array in memory: size $= 8^3 + 4\times(8^2+8^1+8^0)= 804$



$\Longrightarrow$ access via multi-index

FLASH code: Gravity

- Tree Solver: steps to calculate the gravitational potential:
  - communication of the tree:

    1. determine which block-trees to be communicated:

$$\frac{S_{\mathrm{node}}(l)}{D_{\mathrm{min}}} < \texttt{gr\_bhTreeLimAngle}$$

$\Longrightarrow S(l)$: size of the local block-tree at level $l = 0...3$

$\Longrightarrow D_{\mathrm{min}}$: minimum distance to all remote blocks

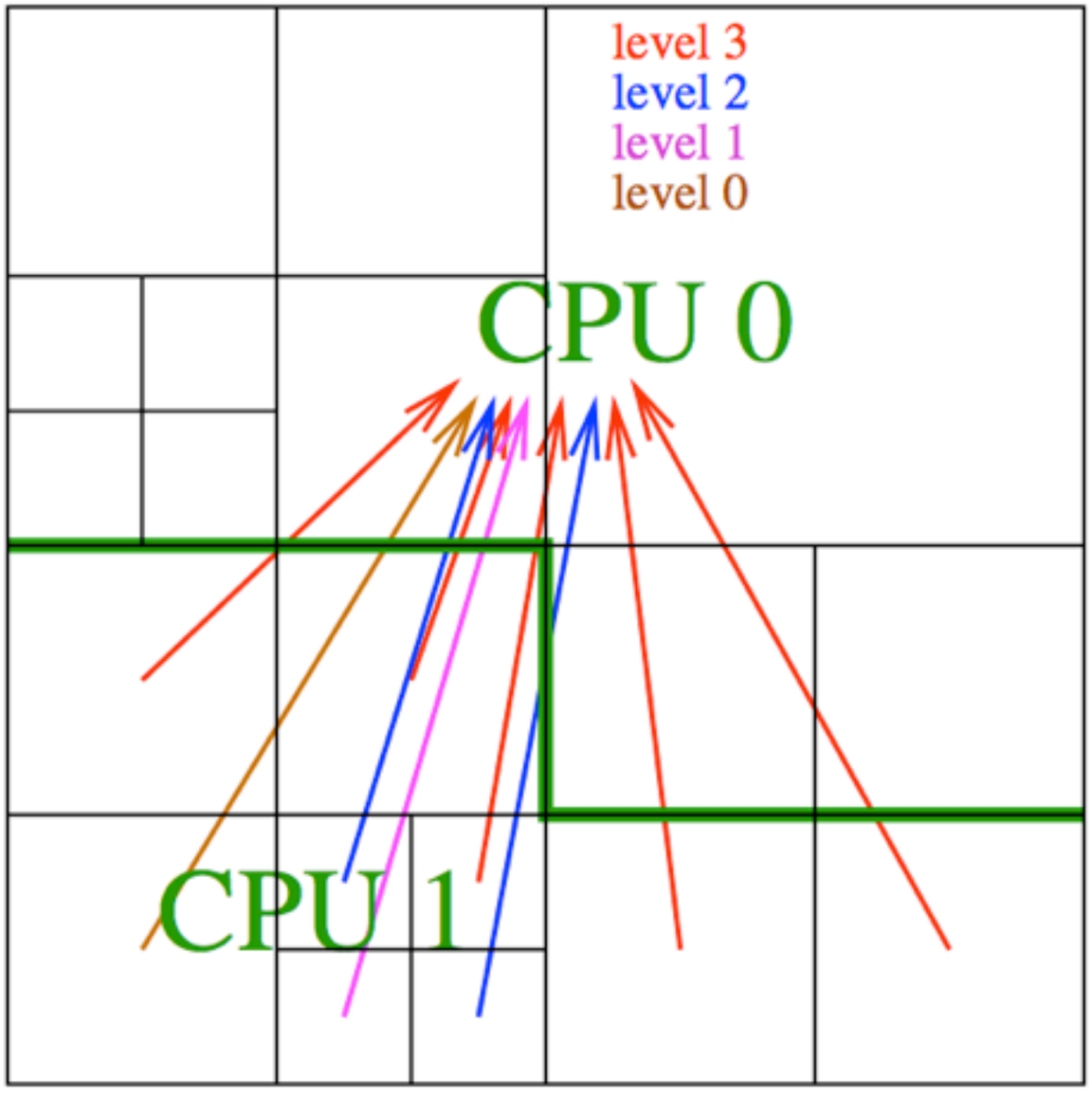
all eight corners of both, the local and the remote blocks, are checked

$\Longrightarrow$ typical values for `gr_bhTreeLimAngle`: 0.5 ... 1.0

FLASH code: Gravity

- Tree Solver: steps to calculate the gravitational potential:
  - communication of the tree:
    1. determine which block-tree levels to be communicated:

$$\frac{S_{\text{node}}(l)}{D_{\min}} < \texttt{gr\_bhTreeLimAngle}$$



    2. communication of block-tree levels
       ⟹ to allocate memory on remote CPUs
       ⟹ all values for a given CPU in a single message

    3. selected block-tree nodes are send to remote CPUs
       ⟹ all block-tree nodes are packed into a single message

FLASH code: Gravity

- Tree Solver: steps to calculate the gravitational potential:
  - calculation of the potential $\Longrightarrow$ the *"tree walk"*

    $\Longrightarrow$ most time consuming part (~ 90 % of tree solver)

    $\Longrightarrow$ fully parallel, no more communication needed

  - walk the tree for local block A

    $\Longrightarrow$ start with refinement level one

    B is the currently processed tree node

  - if B is a parent block

    $\Longrightarrow \mathrm{S}_B/\mathrm{D}_{AB} < \theta_{\mathrm{lim}}$ (S_B: size of B, D_{AB}: min distance to CM_B)

    $\Longrightarrow \Delta\Phi_{A,\mathrm{cell}} = -\,\mathrm{G}\, M_B/\mathrm{D}_{\mathrm{cell},B}$

    else continue with child-blocks of B

FLASH code: Gravity

- Tree Solver: steps to calculate the gravitational potential:

  - if $\boldsymbol{B}$ is a leaf block:

    $\Longrightarrow$ block tree of $\boldsymbol{B}$ is walked separately for each level

    $\Longrightarrow$ block tree consists of nodes $\boldsymbol{N}$

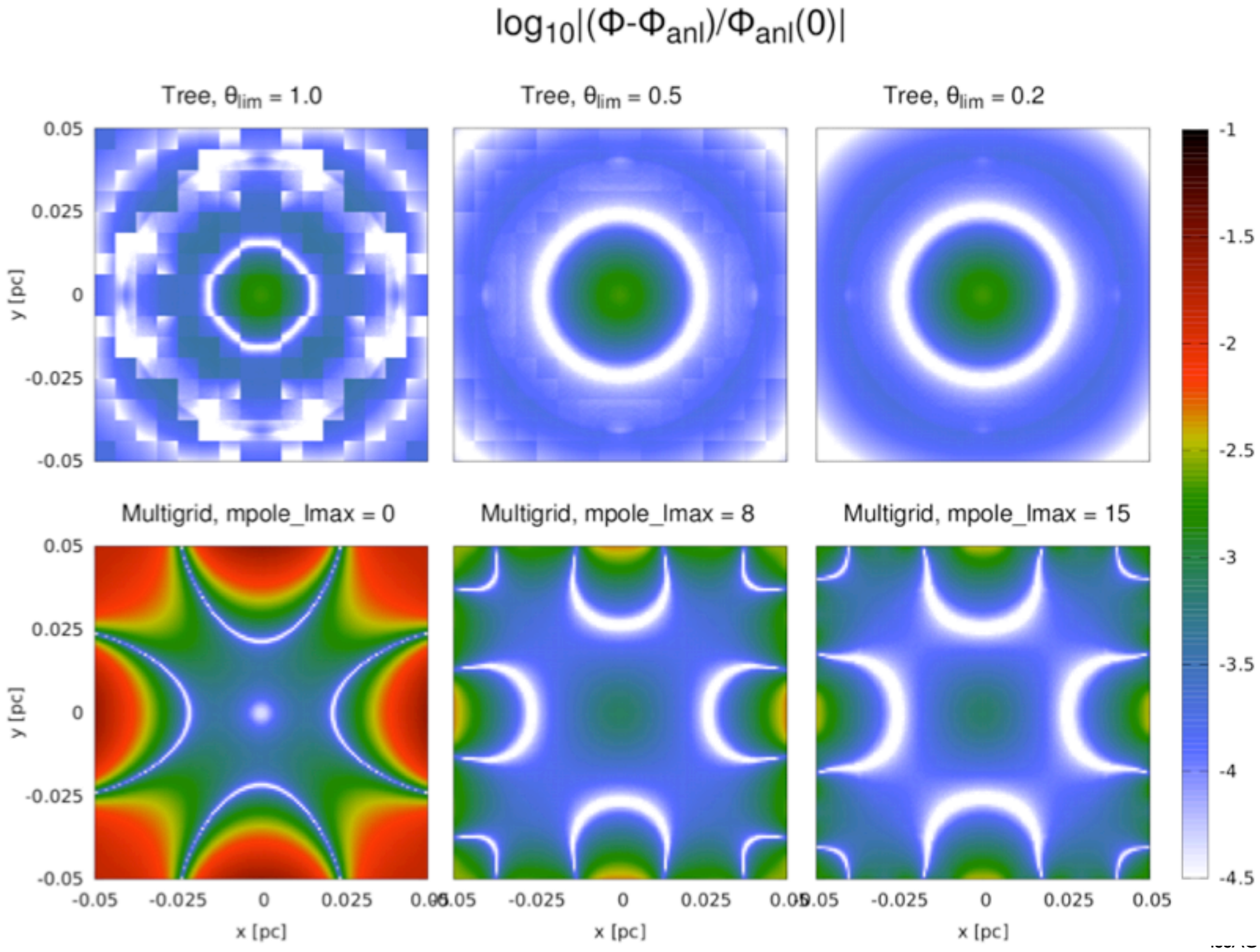
    $\Longrightarrow \mathrm{S}_{\boldsymbol{N}}/\mathrm{D}_{\mathrm{cell},\boldsymbol{N}} < \theta_{\mathrm{lim}}$ ($\mathrm{S}_{\boldsymbol{N}}$: size of node $\boldsymbol{N}$)

    $\Longrightarrow \Delta\Phi_{\boldsymbol{A},\mathrm{cell}} = -\,\mathrm{G}\, M_{\boldsymbol{N}}/\mathrm{D}_{\mathrm{cell},\boldsymbol{N}}$

    else continue with children of $\boldsymbol{N}$

FLASH code: Gravity

- Tree Solver: Test with Bonnor-Ebert sphere



FLASH code: Gravity

- Tree Solver:
  - supported geometry

    $\Longrightarrow$ so far, only 3D cartesian geometry

  - boundary conditions

    $\Longrightarrow$ isolated

    $\Longrightarrow$ periodic → generation of 'Ewald' field values necessary

- Parameters

```
PARAMETER gr_bhTreeLimAngle          REAL      0.5          [ 0 ... ]
PARAMETER gr_bhIlist                 INTEGER   0            [ 0, 1 ]
PARAMETER gr_bhEwaldFieldNx          INTEGER   64           [ 1 ... ]
PARAMETER gr_bhEwaldFieldNy          INTEGER   64           [ 1 ... ]
PARAMETER gr_bhEwaldFieldNz          INTEGER   64           [ 1 ... ]
PARAMETER gr_bhEwaldSeriesN          INTEGER   10
PARAMETER gr_bhEwaldAlwaysGenerate   BOOLEAN   TRUE
PARAMETER gr_bhEwaldFName            STRING    "ewald_field"
PARAMETER gr_bhUseEwaldDecomp        BOOLEAN   TRUE
PARAMETER gr_bhEwaldIsoFac           INTEGER   1000
```

FLASH code: Gravity

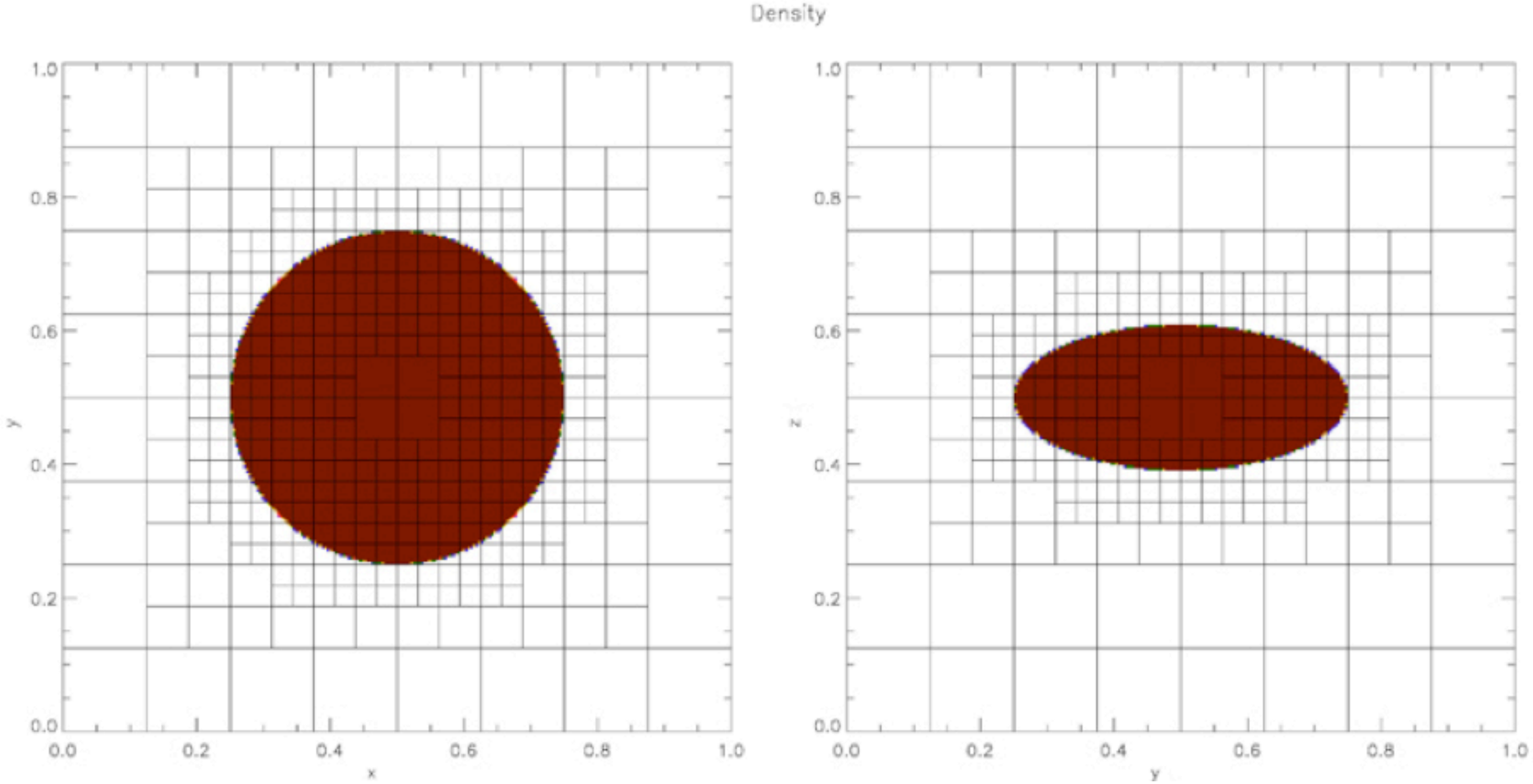
- Tree Solver: the source files in `source/Grid/GridSolvers/BHTree/Wunsch`

```
Grid_solvePoisson.F90
gr_bhBlockRelationship.F90
gr_bhBuildTree.F90
gr_bhBuildTreeBlock.F90
gr_bhComBlkProperties.F90
gr_bhComParentTree.F90
gr_bhData.F90
gr_bhDestroyTree.F90
gr_bhErfc.F90
gr_bhEwald.F90
gr_bhEwaldField.F90
gr_bhExchangeTrees.F90
gr_bhFinalize.F90
gr_bhFindNeighbours.F90
gr_bhGetTreePos.F90
gr_bhGetTreeSize.F90
gr_bhILContrib.F90
gr_bhInit.F90
gr_bhInitTemplates.F90
gr_bhLeafContrib.F90
gr_bhLocalInterface.F90
gr_bhParentContrib.F90
gr_bhPotential.F90
gr_bhPotentialBlock.F90
```

FLASH code: Gravity

- MacLaurin test problem: collapse of a spheroid

`source/Simulation/SimulationMain/MacLaurin`



spheroid with ellipticity $e = 0.9$

FLASH code: Sinks

tasks

- try MacLauren setup with
  - multipole
  - multigrid and
  - BHTree

Poisson solvers

- set up and run a collapse Bonnor-Ebert problem:
  `source/Simulation/SimulationMain/BonnorEbert`

replace `Config` with `/pfs/banerjee/Config_BE`