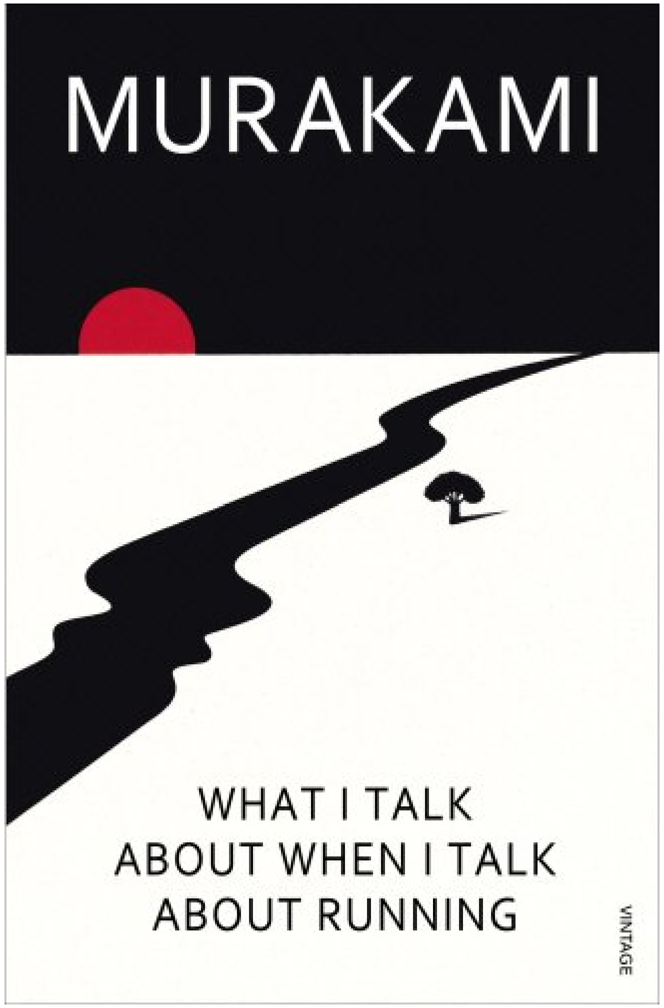
Adaptive Mesh Refinement: Algorithms and Applications

Ann Almgren

Lawrence Berkeley National Laboratory

January 2021

To paraphrase Murakami …



What I Think About When I Think about AMR

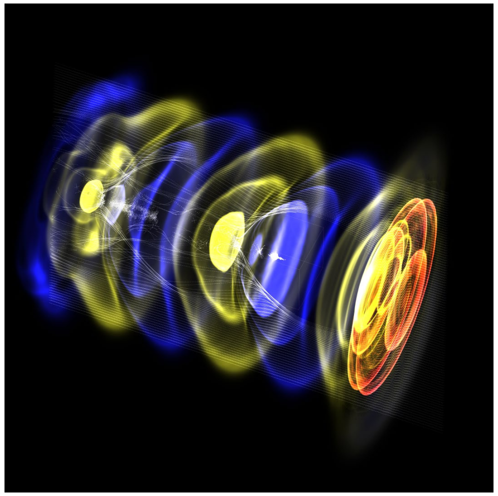
Ann Almgren
Lawrence Berkeley National Laboratory

January 2021

Setting the Stage

Most of the problems we solve today are hard.

Characteristics of these problems are often that they couple multiple physical processes across a range of spatial and temporal scales.

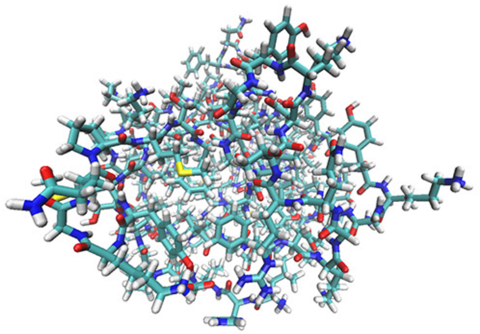

WarpX project: Jean-Luc Vay, PI

Gone are the days of simple physics, simple geometry, single algorithm, homogeneous architectures … ☹

So how do we build algorithms and software for hard multi-physics multi-scale multi-rate problems without starting over every time?

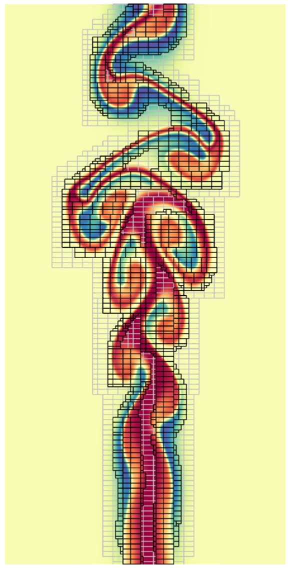
Setting the Stage (p2)

Not all simulations use a mesh



TINKER: https://www.epcc.ed.ac.uk

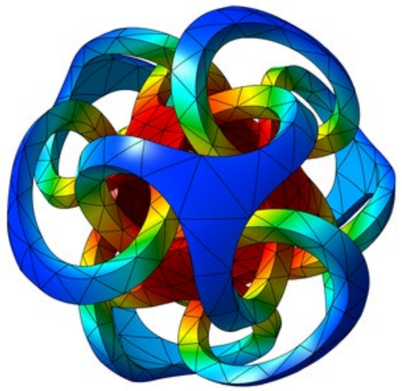
But for those that do, the choice is usually structured vs unstructured.



AMReX: Emmanuel Motheau

Structured:

- Easier to write discretizations
- Simple data access patterns
- Extra order of accuracy due to cancellation of error
- Easy to generate complex boundaries through cut cells but hard to maintain accuracy at boundaries

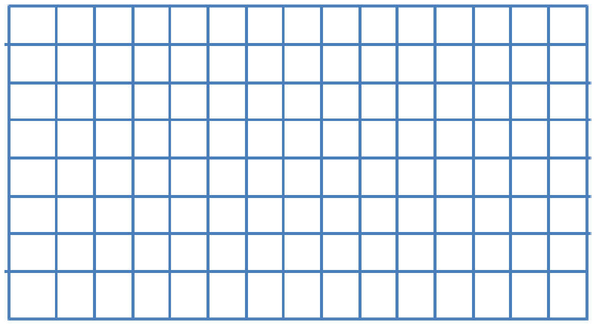


https://ceed.exascaleproject.org/vis/

Unstructured:

- Can fit the mesh to any geometry – much more generality
- No loss of accuracy at domain boundaries
- More "book-keeping" for connectivity information, etc
- Geometry generation becomes time-consuming

Structured Grid Options

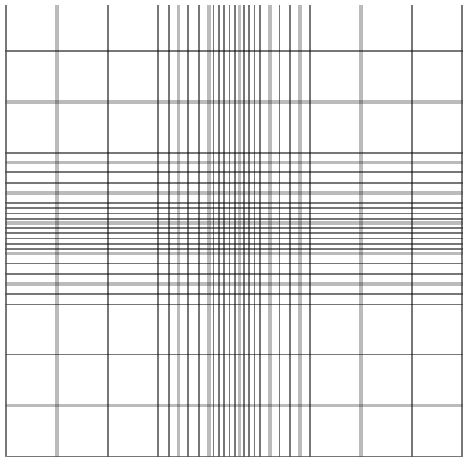


https://commons.wikimedia.org/wiki

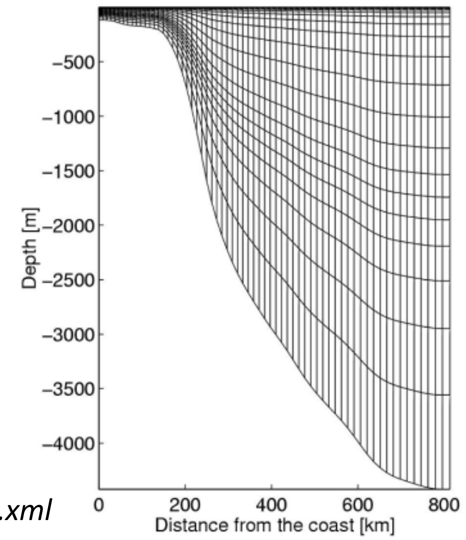
Logically rectangular doesn't mean physically rectangular

Structured with non-uniform cells split pros and cons of structured vs unstructured:

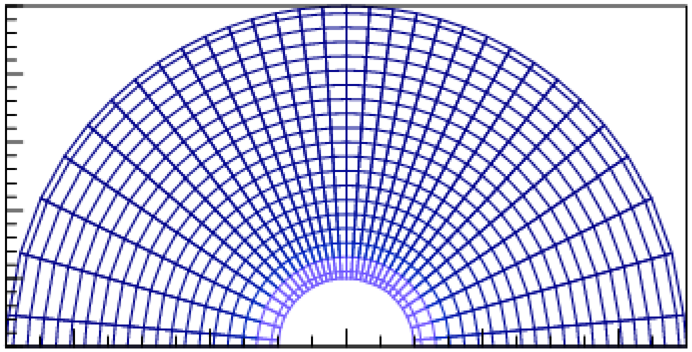
- Can fit (simple) non-rectangular boundaries while still having known connectivity
- Finer in certain regions (mesh refinement)
- Harder to maintain accuracy



http://silas.psfc.mit.edu/22.15/lectures/chap4.xml



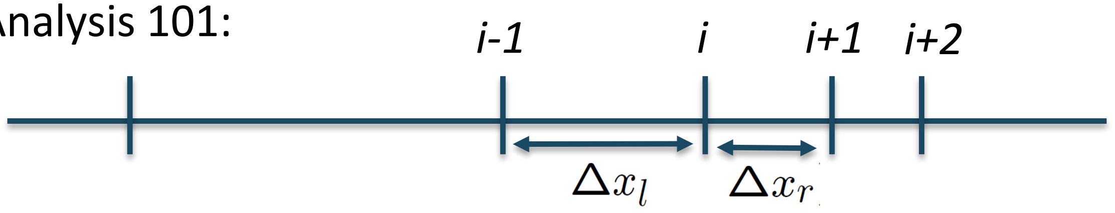
http://www.cfoo.co.za/simocean/modelsroms.php



http://silas.psfc.mit.edu/22.15/lectures/chap4.xml

Why Is Uniform Cell Size Good?

Numerical Analysis 101:



$$\phi_{i+1} = \phi(x_i) + \Delta x_r \phi_x + {}^1\!/\!_2 (\Delta x_r)^2 \phi_{xx} + O({\Delta x_r}^3)$$

$$\phi_{i-1} = \phi(x_i) - \Delta x_l \phi_x + {}^1\!/\!_2 (\Delta x_l)^2 \phi_{xx} + O({\Delta x_l}^3)$$

We might use a **centered difference** as an approximation for a gradient at *i*,

$$\frac{\phi_{i+1} - \phi_{i-1}}{\Delta x_l + \Delta x_r} = \phi_x + {}^1\!/\!_2 (\Delta x_r - \Delta x_l) \phi_{xx} + O(\Delta x^2)$$

Note we only get second-order accuracy if we use constant cell spacing.

Can we confine this error?

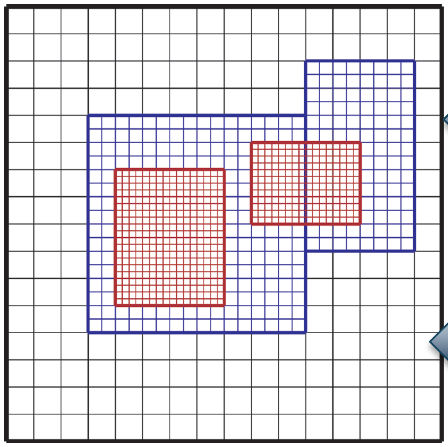
Can We Have the Best Of Both Worlds?

Distorting the mesh is not ideal, but we can't afford uniformly fine grid.

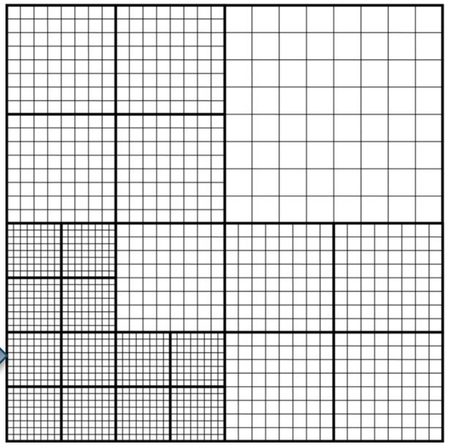
AMR:

- refines mesh in regions of interest
- allows local regularity – accuracy, ease of discretization, easy data access
- naturally allows hierarchical parallelism
- uses special discretizations only at coarse/fine interfaces (co-dimension 1)
- requires only a small fraction of the book-keeping cost of unstructured grids

Example: AMReX



Example: FLASH



Grid sizes

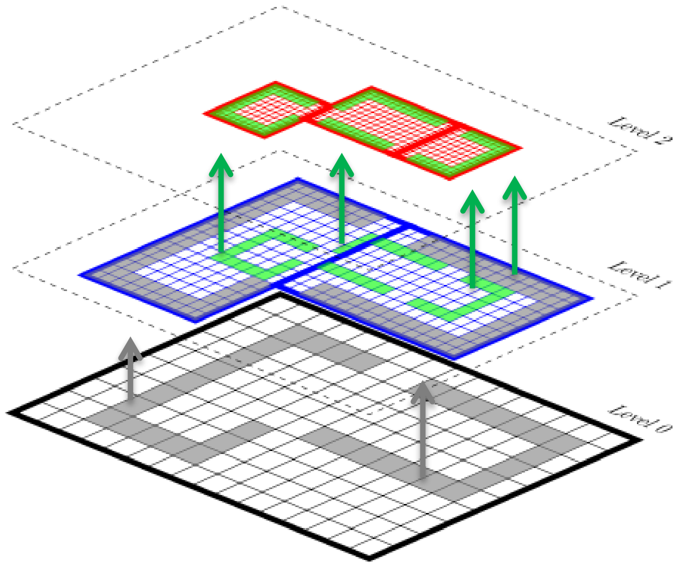
May differ ⇦ ⇨ Same

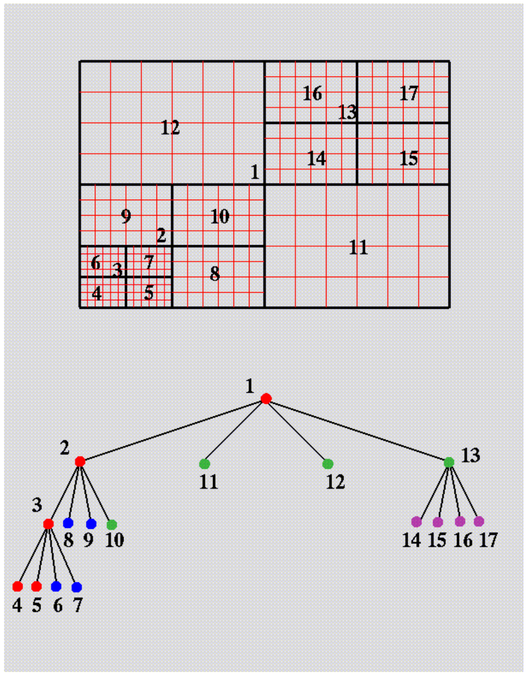
Child grid have unique parent?

No ⇦ ⇨ Yes

https://iopscience.iop.org/article/10.1088/0067-0049/186/2/457)

Level-Based vs OctTree





http://cucis.ece.northwestern.edu/projects/DAMSEL/

Both styles of block-structured AMR break the domain into logically rectangular grids/patches. Level-based AMR organizes the grids by levels; quadtree/octree organizes the grids as leaves on the tree.

What about Time-Stepping

AMR doesn't dictate the spatial or temporal discretization on a single patch, but we need to make sure the data at all levels gets to the same time.

The main question is:

To subcycle or not to subcycle?

Subcycling in time means taking multiple time steps on finer levels relative to coarser levels.

Non-subcycling:

- Same dt on every grid at every level
- Every operation can be done as a multi-level operation before proceeding to the next operation, e.g. if solving advection-diffusion-reaction system, we can complete the advection step on all grids at all levels before computing diffusion

Subycling:

- dt / dx usually kept constant
- Complete all the operations on one level before advancing the next level -- requires separation of "level advance" from "synchronization operations"
- Can make algorithms substantially more complicated

So Why Subcycle?

Consider linear 1-d advection with a constant velocity field u = 1.

Set scalar phi = 1 from x = 0.5 to x = 0.59375, elsewhere 0.

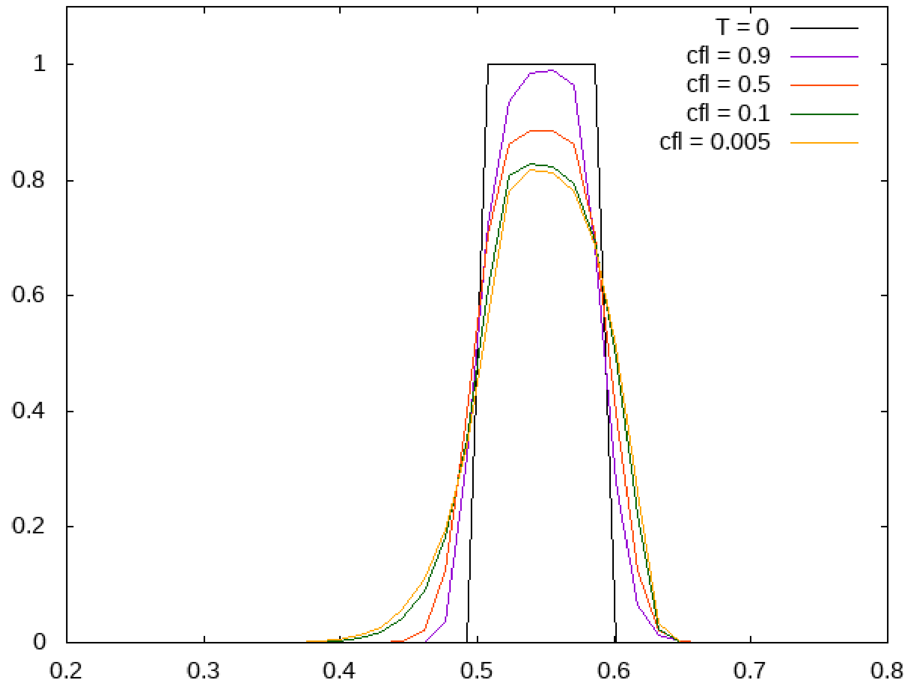
Run with a second-order advection scheme from T = 0 to T = 1 with dt = cfl*dx / u.

Note that without subcycling, if we run with cfl = 0.9 on the finest level, the effective cfl on coarser levels is
0.9 / 2^{fine level – crse level}

If all the features of interest are on the finest level, this is a non-issue.

This may constrain our flexibility in refinement criteria

Phi at T = 0, T = 1 with different cfl



Domain is unit cube; periodic bc's; dx = 1/64

So Why Subcycle? (p2)

Consider a simulation that zooms in using n refined levels (factor 2), each with the same number of cells N as level 0 (e.g. refines 1/8 of the coarser level in 3D).

To advance the solution from T to T + (delta T)_0 would require 1 time step based on the coarse grid resolution.

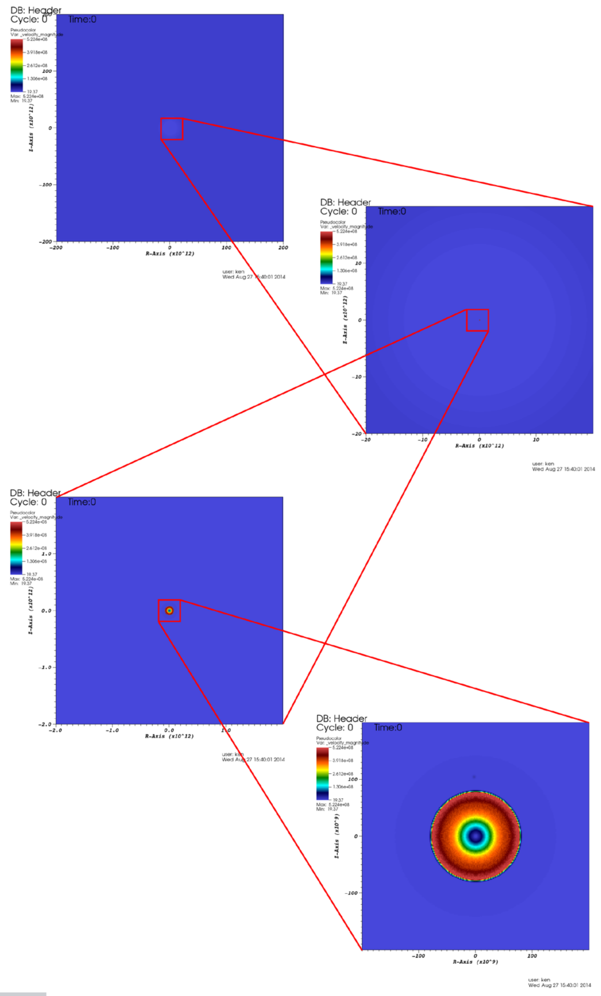
Assume (delta T)_{n} = (delta T)_{0} / 2^n.

Subcycling reduces the total amount of (non-sync) work

Levels	Non-subcycling	Subcycling	Ratio
n = 0	1 * 1 = 1	1 * 1 = 1	1
n = 1	2 * 2 = 4	1 + 2*1 = 3	4/3 = 1.3
n = 2	4 * 3 = 12	3 + 4*1 = 7	12/7 = 1.7
n = 3	8 * 4 = 32	7 + 8*1 = 15	32/15 = 2.1
n = 4	16 * 5 = 80	15 + 16*1 = 31	80/31 = 2.6
n = 5	32 * 6 = 192	31 + 32*1 = 63	192/63 = 3.0
n = 6	64 * 7 = 448	63 + 64*1 = 127	448/127 = 3.5

Power of 10 (CASTRO)

Each close-up is by a factor of ten



Why Not Subcycle?

- The work estimate on the previous slide is misleading because it only counts "single-level work." Subcycling requires potentially complicated synchronization between levels (some synchronization still required for non-subcycling)

- Subcycling may not be possible:
  - If time advance paradigm is to loop over all leaves on the (oct-)tree, subcycling is not an option
  - Subcycling requires being ***able*** to advance the solution on coarse regions covered by fine grids – this may not be possible

- Improved load balancing can off-set the cost of doing more total work if not enough grids at some intermediate levels

Take-away re time-stepping

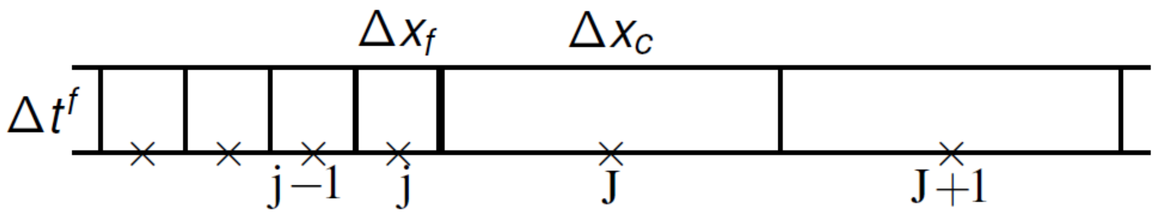
- Compromise position is to subcycle between some levels but not others – we call this "optimal subcycling" – can be more efficient than either/or
- There are options that iterate over levels in more complex way to generate higher-order solution
- Software that supports both options gives the user the chance to optimize

Regardless of the global time-stepping procedure, coarse and fine data need to communicate

1D Hyperbolic Example

Consider U_t + F_x = 0 discretized with an explicit finite volume scheme

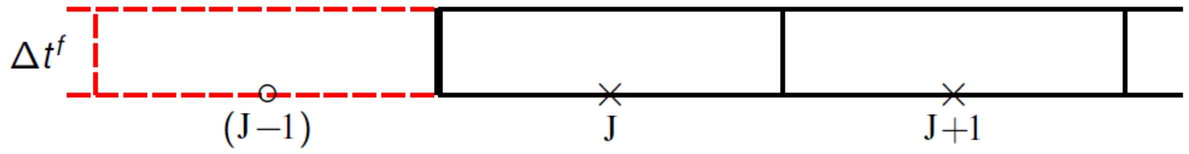
$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = \frac{F_{i-1/2}^{n+\frac{1}{2}} - F_{i+1/2}^{n+\frac{1}{2}}}{\Delta x}$$



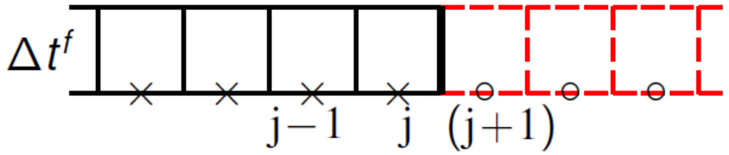
Advancing the solution level by level:

One can advance the coarse grid

Red = averaged down from fine



then advance the fine grid



Red = interpolated from coarse

Here we consider using the same dt at both levels

What is true about the composite solution that may not be satisfied if we advance the levels separately?

- Is the coarse data the average of fine data where possible?
- Are the fluxes that we used to update the solution consistent?

Synchronization = correcting the mismatches

The answer, of course, is "not yet". To complete the advance we must

- make coarse data be average of fine data where possible
- use fluxes from fine level where possible

We achieve the first by averaging the fine data onto the coarse grid.

We achieve the second by **refluxing:**

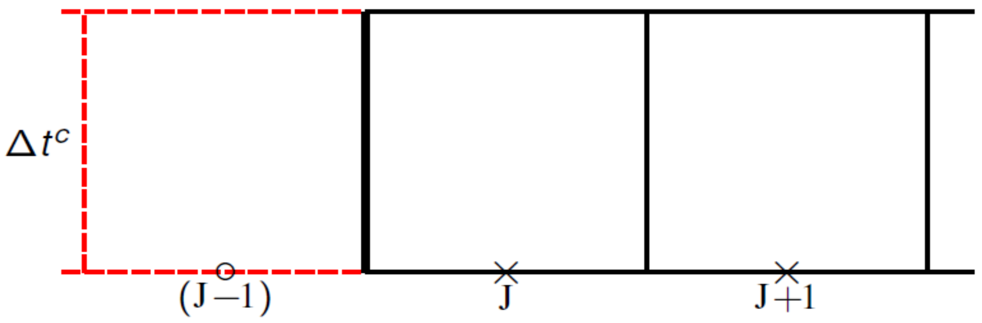
$$\Delta x_c U_J^{n+1} := \Delta x_c U_J^{n+1} - \Delta t^f F_{J-1/2}^c + \Delta t^f F_{j+1/2}^f$$

Observation: in the end we have exactly the same solution as if we defined the correct flux at the coarse/fine boundary before using it on either level.

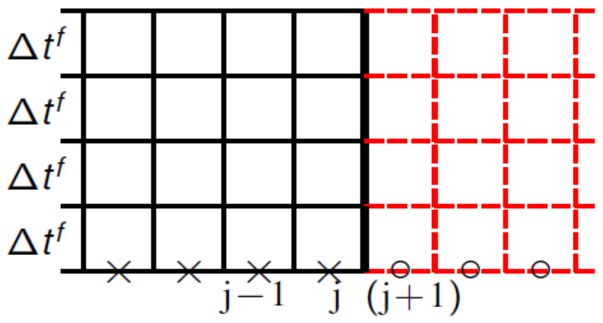
What happens if we subcycle?

To subcycle in time we advance the coarse grid with Δt^c

Red = averaged down from fine



and advance the fine grid multiple times with Δt^f.



Red = interpolated from coarse

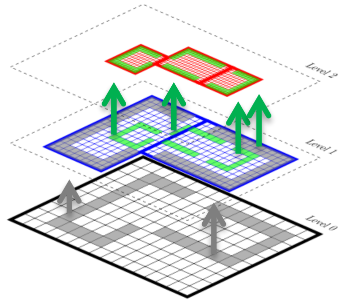
The refluxing correction now must be summed over the fine grid time steps:

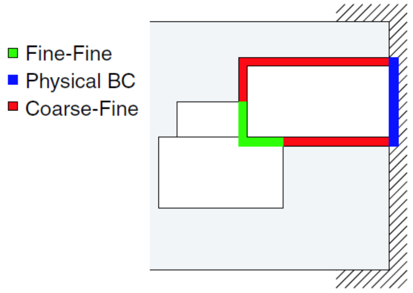
$$\Delta x_c U_J^{n+1} := \Delta x_c U_J^{n+1} - \Delta t^c F^c_{J-1/2} + \sum \Delta t^f F^f_{j+1/2}$$

This makes subcycling look pretty easy

Recursive integration with subcycling:

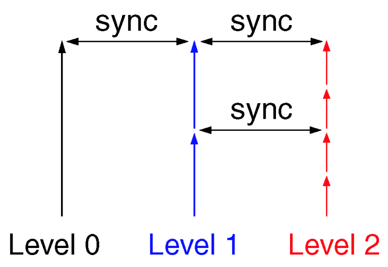
- Integrate each patch at level n without seeing finer levels
- Fill ghost cells for next finer level, interpolating in space and time where necessary
- (Keep track of fluxes used at coarse/fine boundaries)
- Integrate level n+1 grids for r time steps





When the coarse and data are at the same time,

- Average the fine data onto coarse levels
- “Reflux” → update coarse cells next to coarse/fine boundary so it is as if we used the fine fluxes from the beginning
- Note that refluxing only changes the solution immediately adjacent to the fine grids



Extend this reasoning to elliptic equations

Suppose we want to solve Laplacian(phi) = RHS

RHS on fine grid = +/- ;
RHS on coarse grid = 0.

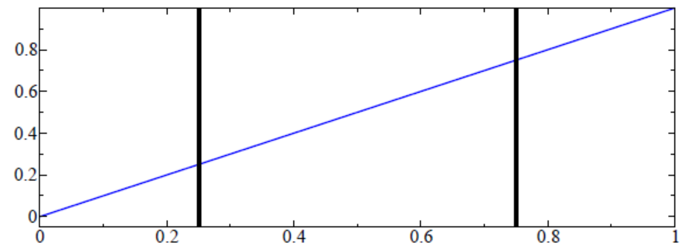
Classic one-way coupling (eg Enzo):

- Solve on coarse level
- Solve on fine level using Dirichlet bc's from coarse level

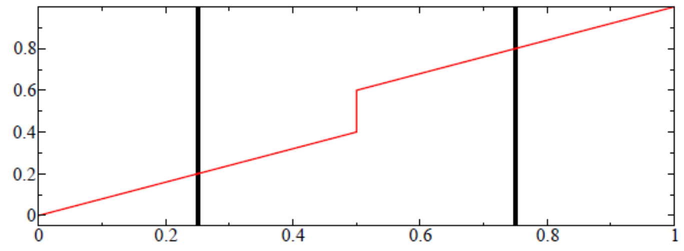
Using our paradigm, we must first identify the mismatch. Since this is a second-order operator, the composite solution must satisfy both Dirichlet and Neumann matching conditions

- We have a Neumann mismatch

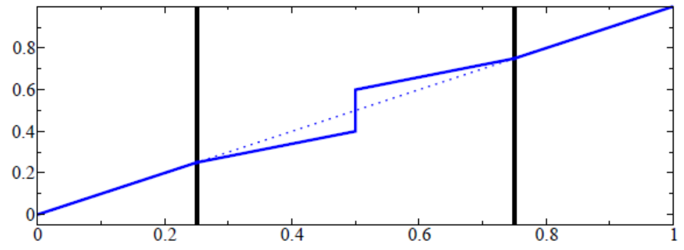
What is the analogous reflux operation?



Uniform coarse



Uniform fine

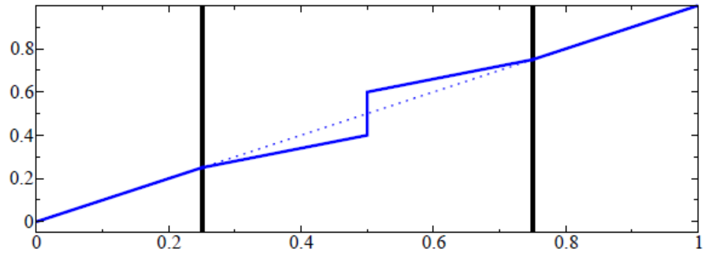


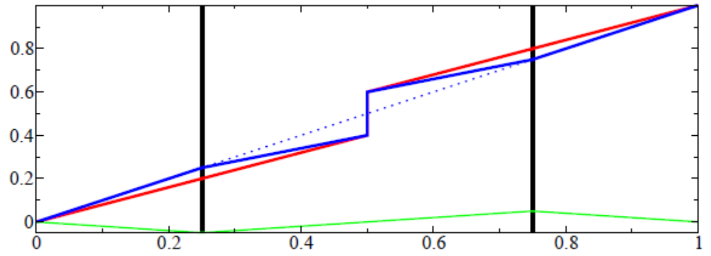
Coarse + fine with no sync

Synchronization for Elliptic Equations

How to "reflux":

- The Neumann mismatch at the coarse-fine boundaries become the source term for a "sync solve".
- Compute the correction to the solution by solving a new elliptic equation and add to the solution on the coarse and fine level – everywhere!





Green = correction;
red = composite solution

Takeaways:

- The correction is not always local.
- The synchronization operation depends on the original operation:
  - Explicit local reflux for explicit advance of hyperbolic system
  - Elliptic correction equation for solve of elliptic system

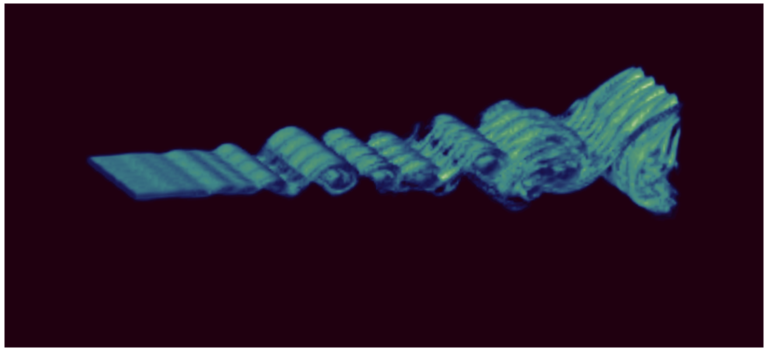
Fast-forward to incompressible Navier-Stokes (1998)

$$U_t + (U \cdot \nabla)U = \frac{1}{\rho}(-\nabla p + \mu \nabla^2 U + H_U),$$

$$\rho_t + \nabla \cdot (\rho U) = 0,$$

$$c_t + (U \cdot \nabla)c = k\nabla^2 c + H_c,$$

$$\nabla \cdot U = 0,$$



Mismatches:

(M.1) The data at level ℓ that underlie the level $\ell+1$ data are not synchronized with the level $\ell+1$ data.

(M.2) The composite advection velocity computed from the MAC projection, defined as the time-averaged (over a level ℓ time step) level $\ell+1$ advection velocity on all level $\ell+1$ faces, including the $\ell/(\ell+1)$ interface, and the level ℓ advection velocity on all other level ℓ faces, does not satisfy the composite divergence constraint at the $\ell/(\ell+1)$ interface. This mismatch results in spatially constant advected quantities with no source terms not remaining constant.

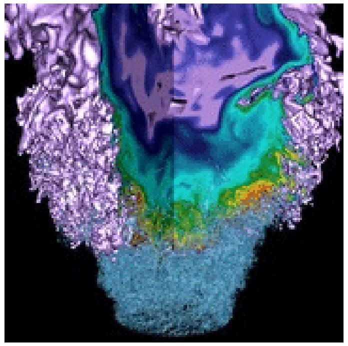
(M.3) The advective and diffusive fluxes from the level ℓ faces and the level $\ell+1$ faces do not agree at the $\ell/(\ell+1)$ interface, resulting in a loss of conservation.

(M.4) The composite new-time velocity, defined as the level $\ell+1$ new-time velocity on all level $\ell+1$ cells, and the level ℓ new-time velocity on all level ℓ cells not underlying level $\ell+1$ grids, does not satisfy the composite divergence constraint at the $\ell/(\ell+1)$ interface.

Synchronization:

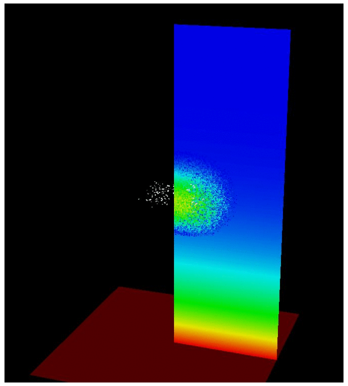
- Correct for the advective mismatch…
- Feed that correction into the source term for the diffusive mismatch …
- Feed those corrections into the source term for the projection correction …

Fast-forward from 1998 ..



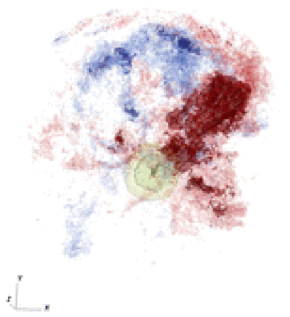
AMR has a long history in compressible astrophysics and other compressible phenomena.

Extensions of AMR usage include



- Low Mach number Combustion – heat release may look very different on coarse and fine levels
- Low Mach number astrophysics – 1-d background state plus perturbational solution
- Moist atmospheric modeling
- Solid mechanics, e.g. microstructure evolution

Flows with particles add complexity when particles and grids interact

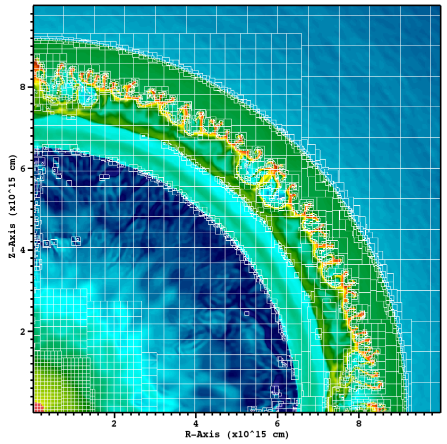


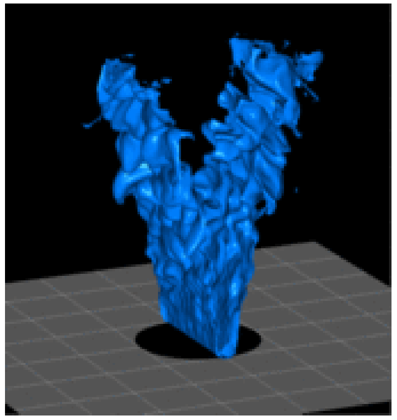
Especially interesting ways to use AMR include AMAR (Adaptive Mesh and **Algorithm** Refinement) …

If you can solve one …

If we know how to solve one type of low Mach number flow – which we do! -- we can re-use most of the existing software. Elements of an AMR code for low Mach number simulation may include

- Projection method formulation (including advection, diffusion, reactions)
- Variable density Poisson solve for updating (perturbational) pressure and velocity
- ODE integrator for reaction network
- Adaptive mesh refinement – to address the range of spatial scales

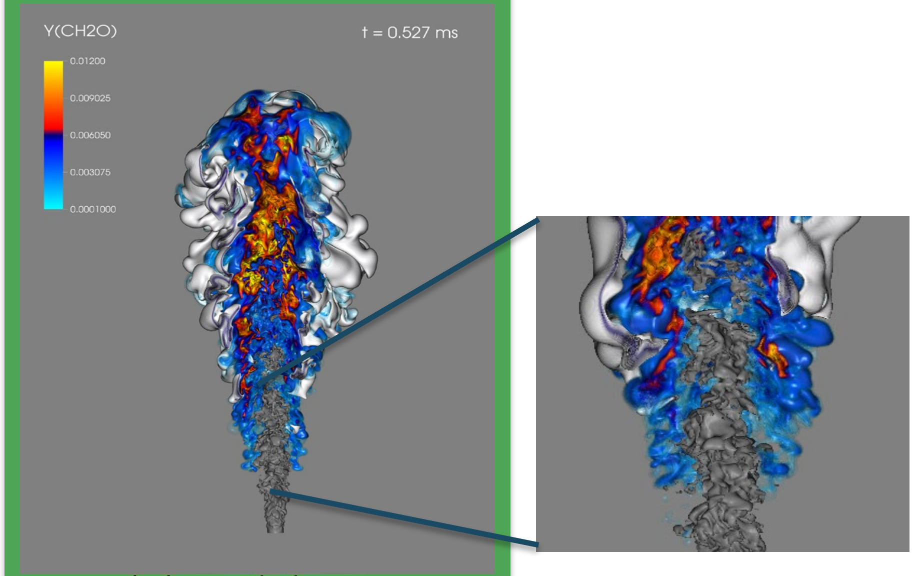




Combustion Modeling using PeleLM

PeleLM (formerly LMC) is a low Mach number combustion code that includes convection, species and thermal diffusion, reaction networks.

Applications range from detailed flame studies to lab-scale flames



Turbulent n-dodecane jet, computed with adaptive mesh and detailed chemistry using PeleLM
D. Dalakoti, et al 2019.

Colors = formaldehyde
White/grey = OH

What is hard in low Mach combustion?

- Time integration of reaction network -- Strang splitting less desirable with larger dt
- Evolution of background state p0(t) on adaptive hierarchy in closed chambers

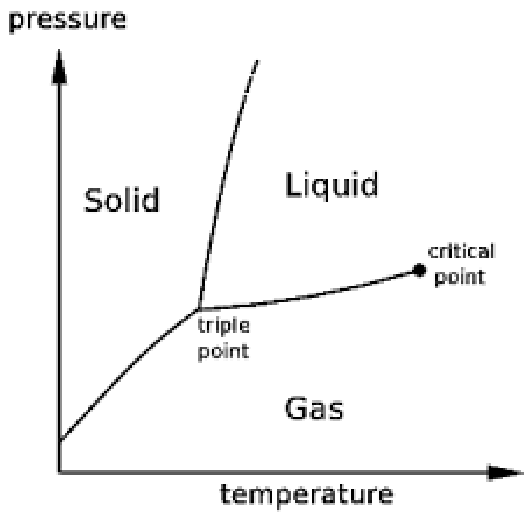
Note that same code framework can be extended to astrophysical nuclear burning studies

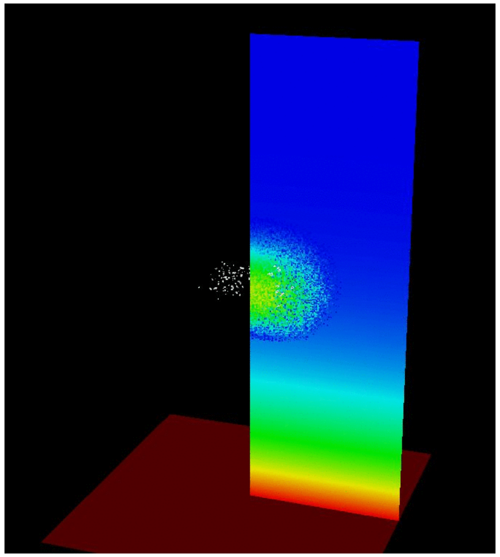
Dalakoti, Hawkes, Day & Bell; 26th ICDERS 2017

Moist atmospheric Flows

We can model moist atmospheric flows in a stratified background

- **Algorithmic** complication:
  - Clausius-Clapeyron gives equilibrium relationship for mixture of dry air, water vapor and liquid water
  - Divergence constraint needs heat release which is based on evaporation rate





Astrophysical Convection using MAESTRO

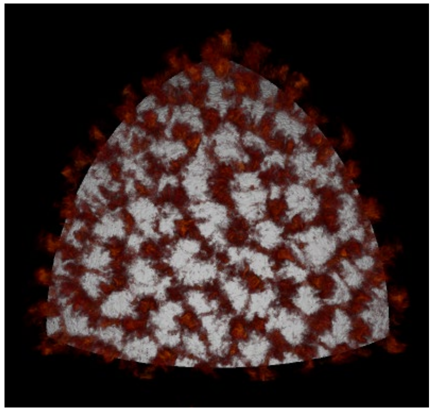
Simulation of low Mach number astrophysical flows looks similar to that of atmospheric flows – but adds self-gravity, nuclear reaction networks. We either zoom in on center or surface of star, depending on application.

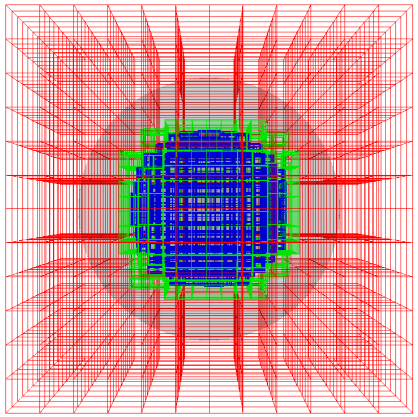
Applications include:

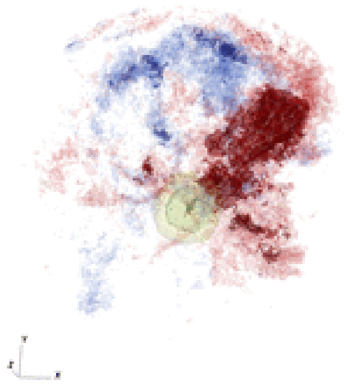
- two types of Type Ia progenitors
    - Chandrasekhar mass
    - sub-Chandra
- X-ray bursts,
- convection in massive stars.

Algorithmic complications include:

- Strang splitting less desirable with larger dt
- Evolution of background state (p0 a function of z instead of t)







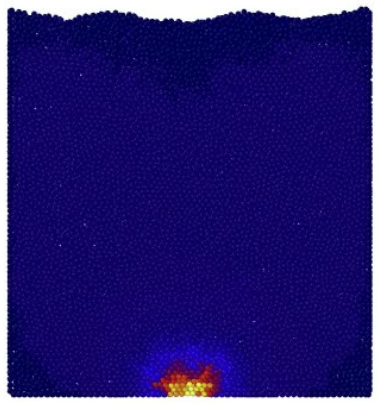
Multiphase Flows

Another interesting low Mach number flow – with interesting convective features – is multiphase flow in which solid particles occupy a volume fraction and exchange momentum with the ambient gas via drag terms.

$$\frac{\partial}{\partial t}(\varepsilon_g \rho_g) + \boldsymbol{\nabla} \cdot (\varepsilon_g \rho_g \boldsymbol{u}_g) = 0 \ ,$$

$$\frac{\partial}{\partial t}(\varepsilon_g \rho_g \boldsymbol{u}_g) + \boldsymbol{\nabla} \cdot (\varepsilon_g \rho_g \boldsymbol{u}_g \boldsymbol{u}_g) = -\varepsilon_g \boldsymbol{\nabla} p_g + \boldsymbol{\nabla} \cdot \boldsymbol{T}_g + \varepsilon_g \rho_g \boldsymbol{g} - \sum_{m=1}^{M} \boldsymbol{I}_{gm} \ .$$

where ε_g represents the volume fraction of gas


MFIX-Exa project

AMAR: different physics at different levels

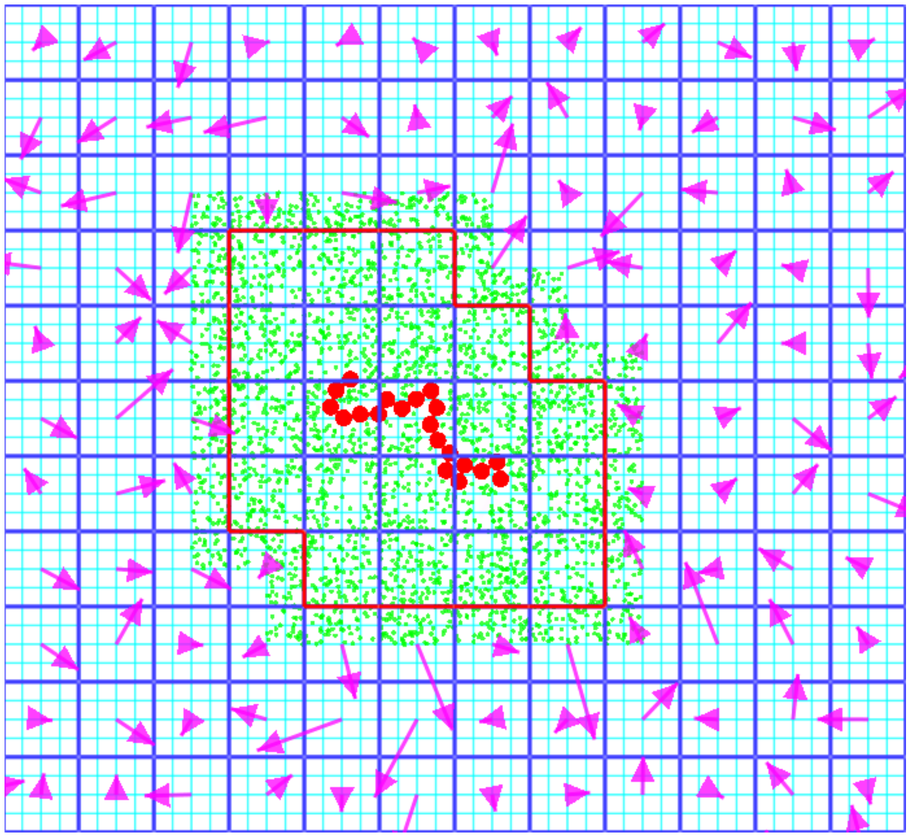
Consider a DSMC algorithm in a small region of the domain coupled with a fluctuating hydrodynamics (FHD) algorithm:

Level 0 covers the entire domain – we solve the stochastic PDEs for FHD on all coarse cells

Level 1 covers a small region (bounded by orange) – we solve the DSMC equations

Coupling:

- Interpolation replaced by statistical realization (mesh->particles)
- Averaging is now particles->mesh
- Refluxing – continuum flux replaced by particle flux



Red: bead model for polymer
Green: DSMC particles
Purple coarse mesh: for solving FHD
Pink arrows: velocity field on the mesh

Courtesy of John Bell, Aleks Donev and Alej Garcia

AMR Requires Good Software Support

There are a number of AMR software packages available –

They all

- Provide data containers for blocks of data at different resolutions
- manage the metadata – same-level and coarse-fine box intersections
- manage re-gridding (creation of new grids based on user-specified refinement criteria)

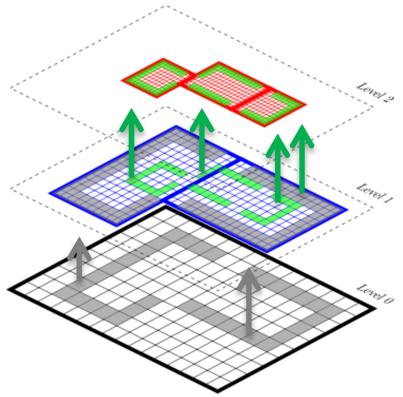
They differ on:

- what types of data they support (cell/face/node-centered)
- what types of time-stepping they support (many are no-subcycling only)
- whether they support separate a "dual grid" approach
- what degree of parallelism do they support? MPI only, MPI+X (what X?)
- what task iteration support – asynchronous, fork-join, kernel launching…?
- how flexible is the load balancing?
- what additional features – e.g. AMR/GMG solvers?

Overview of AMReX

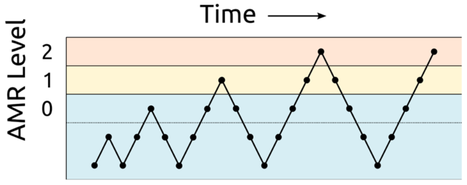
AMReX is a software framework for massively parallel block-structured AMR applications

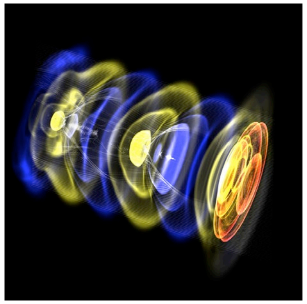
- Written in **C++11** (with an option for using Fortran interfaces and calling Fortran kernels)
- Supports **parallelism through MPI, MPI+X (+X)**
  - Multi-core: "MPI over grids, OpenMP over tiles" – static vs dynamic scheduling
  - Hybrid w/ GPUs: "MPI over grids, CUDA (or HIP or DPC++) on GPUs"
- Multilevel **mesh** data and iterators
- Multilevel **particle** data and iterators
- **Explicit & implicit** single- and multi-level mesh operations
- **Particle-particle** operations and **particle/mesh** operations



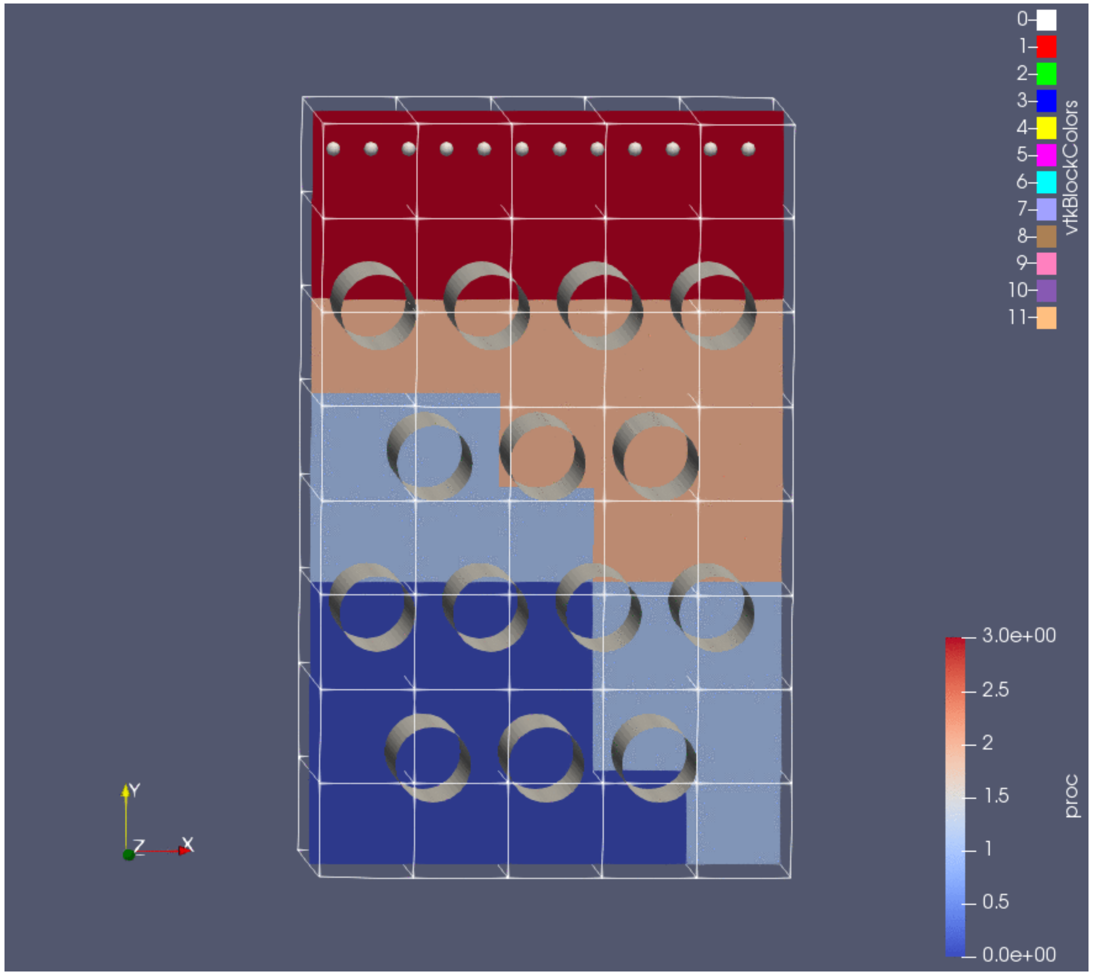
Overview of AMReX (p2)

- **Embedded boundary** (cut-cell) representation of geometry
  - option to store level set / distance function on mesh
- **Geometric multigrid solvers** for solution of parabolic and elliptic systems
  - options to call hypre/PETSc AMG solvers
  - consolidation & aggregation on coarsening
  - EB-aware options
- Support for multiple **load balancing** strategies (including dual grid approach for separate domain decomposition for mesh vs particles)
- **Native I/O format** – supported by Visit, Paraview, yt





Load Balancing Depends on the Application



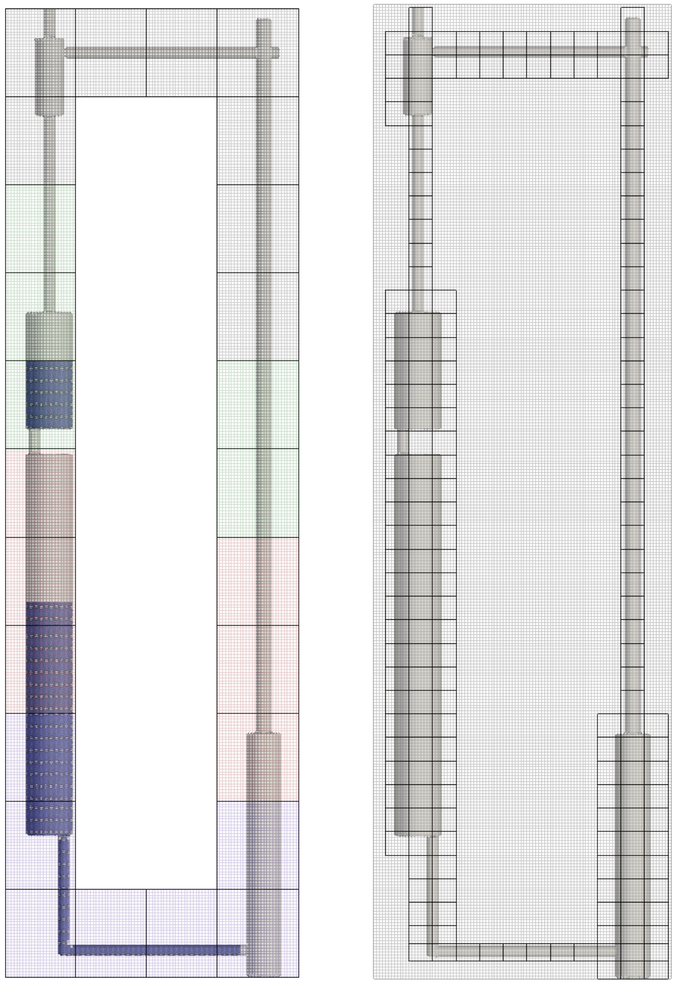
We could load balance based on:

- Number of grid cells
- Number of particles
- Number of "cut cells"

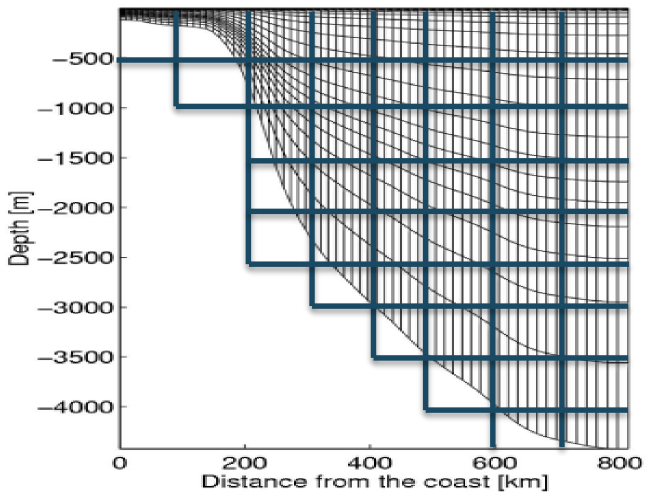
We could try to be really smart and figure this out ahead of time…

Or we could use estimates from past time steps

Grid Pruning Can Save Memory and Work



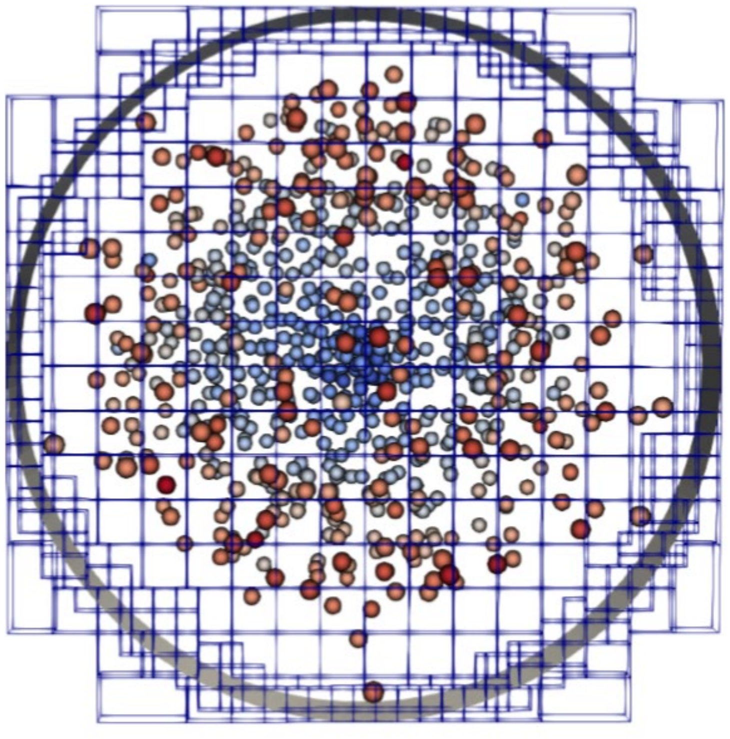
Once you have software flexible enough to support AMR, there are other useful features, such as the ability to "prune" the grids so as to not waste memory or MPI ranks – can still use rectangular cells in non-rectangular domain.



“AMR for One” does not have to mean “AMR for All”

For example, in the MFiX-Exa code, we define a level set that holds the distance to the nearest wall . The level set is only used by the particles to compute particle-wall collisions.

We refine the mesh on which the level set is defined in order to capture fine geometric features … but the particles and fluid are both defined on the coarser mesh only.



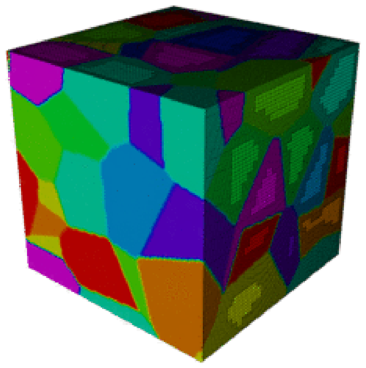
Particles, particle mesh, and level set mesh at the bottom of a cylinder in an MFiX-Exa simulation.

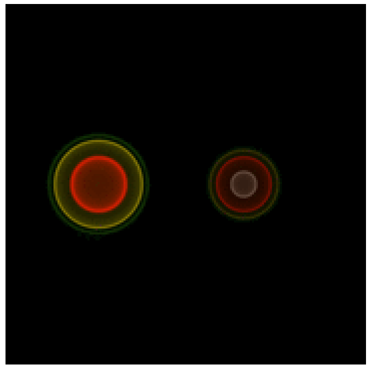
AMReX

Supported by:

- ECP Block-Structured AMR Co-Design Center
- FASTMath SciDAC-5 Institute

Part of xSDK software releases.


Brandon Runnels, Vinamra Agrawal et al


Max Katz, Michael Zingale et al

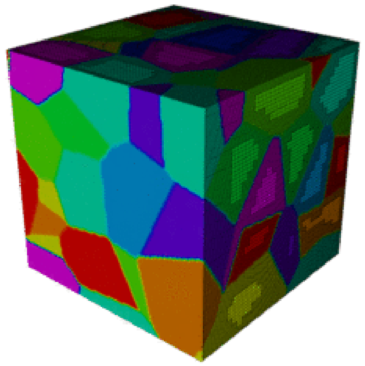
Brayden Roque, Hsiao-Chi Li, and Ryan Houim

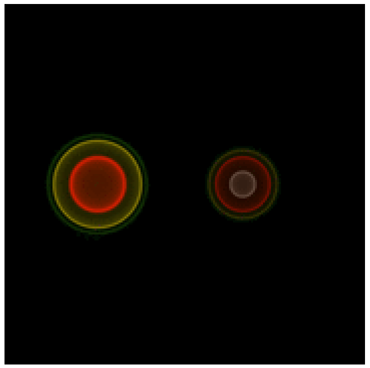
Thank you!

If you're interested in looking at the software:
https://www.github.com/AMReX-Codes/amrex

For some movies based on AMReX:
https://amrex-codes.github.io/amrex/gallery.html

If you're interested in the documentation:
https://amrex-codes.github.io/amrex


Brandon Runnels, Vinamra Agrawal et al


Max Katz, Michael Zingale et al

Brayden Roque, Hsiao-Chi Li, and Ryan Houim