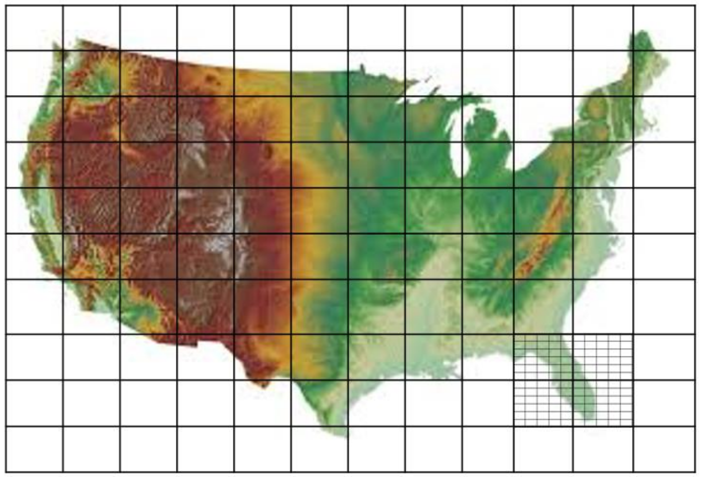
Adaptive Mesh Refinement in the Age of Exascale Computing

Ann Almgren

August 2024

Defining "Adaptive Mesh Refinement (AMR)"



Mesh refinement generically refers to finer resolution in some spatial region(s) than others.

Sometimes we use "nesting" instead … and sometimes "coupling" also implies change of resolution. These tend to imply statically defined regions.

- **adaptive** = "solution-adaptive" so possibly changing in time
- "block-**structured** refinement" = at each level the mesh is logically structured (but we can still have map factors and such)
- "**block**-structured refinement" = refinement always occurs in "blocks" as opposed to cell by cell refinement

Block-structured AMR has a long history in DOE

In the world of DOE math / computing (i.e. ASCR),

- First developed for compressible flows – hyperbolic conservation laws (1980s)
  - 2-way nesting was straight-forward = “average down” + local “reflux”
  - Computational astrophysics / cosmology were early adopters

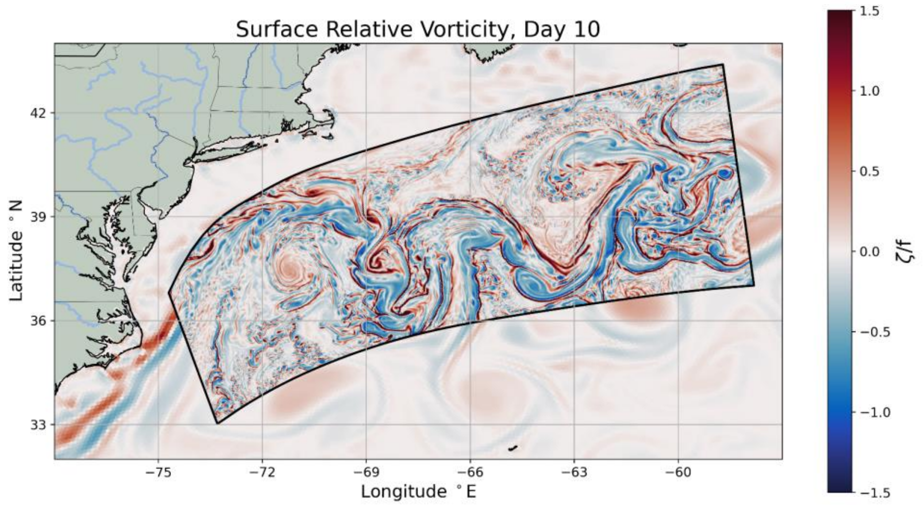
- AMR for elliptic problems came later (1998-ish)
  - 2-way synchronization across levels requires another elliptic solve (and “reflux” no longer a local operation)
  - Extension to parabolic (e.g. implicit diffusion) was straightforward
  - Extensions to other low-Mach number flows (e.g. anelastic, reacting) wasn’t always straightforward but we now know how to use AMR for a number of different types of flows

In this development path, the focus has always been on two-way coupling, but for relatively straightforward systems of equations.

But there are still challenges

This all gets more interesting, and more challenging, when

- physics representations vary between scales, so the equations to be solved may not be identical at different levels of resolution
- we use different codes with different discretizations at different levels



Example of ROMS/MPAS-O coupling from SEAHORCE (Rob Hetland, PI) project (courtesy of Kyle Hinson)

Software for AMR: Status Update

We have the software experience and capability to support simulations using AMR. We are also getting much better at efficient coupling of codes, whether through external couplers or using more intrusive approaches.

“**In the age of exascale computing**” means

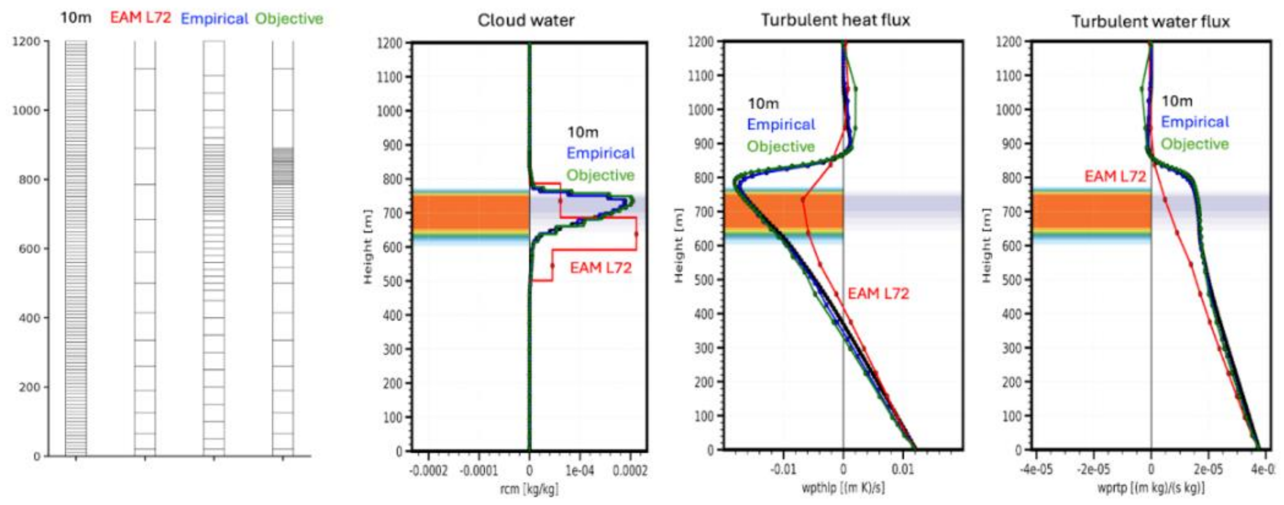
- we have to care about having good software support for simulations on multiple regions at multiple levels – “roll your own” is no longer feasible

- we can’t assume a homogeneous architecture (i.e. a code that works on only on CPU or only with one flavor of GPU won’t meet our needs)

- we can’t assume the architecture will stay the same over time – and the science applications shouldn’t be trying to face this alone

Additional Challenges

The challenges we should be focusing on now are how to couple across scales and across codes (methods). In addition, it is not always obvious what the refinement criteria should be.



This example from the PAESCAL project (Hui Wan, PI) demonstrates the use of refinement in the 1-D CLUBB model; it is straightforward to see where the refinement should be once we know the solution (from a fine calculation), but not obvious a priori.

Takeaways

- Leverage the AMR software and the expertise that already exists – but do so as part of a partnership / collaboration. Trying to use software thrown over the fence without longterm support is an exercise in frustration.

- We need domain knowledge to understand what models to use at what resolutions, and how to couple those models across length and time scales. Focusing on these issues both allows us to have more efficient and powerful simulation codes, but also increases our understanding of the multiscale phenomena we're studying.

- Deciding when/where to refine also requires both domain knowledge and good understanding of computational costs.

- We don't have to do it all at once.   It is possible to improve one component (e.g. a single1D submodel) at a time while we also figure out how to effectively use mesh refinement more broadly.