GRChombo: Adaptive Mesh Refinement and Tagging Criteria

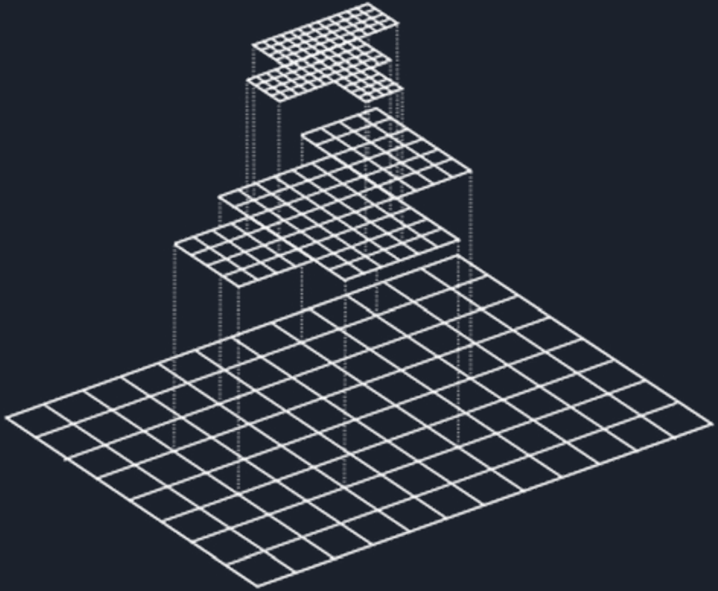
Miren Radia, DAMTP, University of Cambridge

Conventions

- *Important/technical term*
- `GRChombo base parameter`
- `C++ or pseudocode`

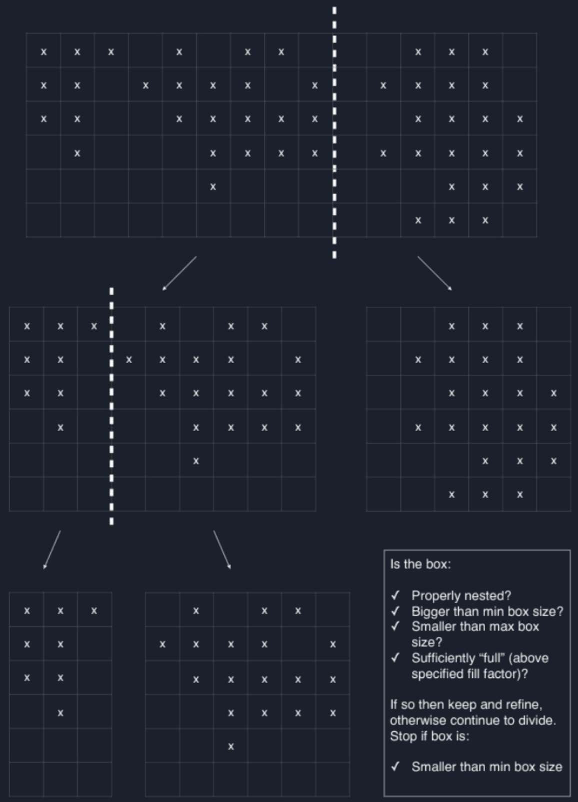
Grid Structure

- [GR]Chombo uses a hierarchy of *properly nested* block-structured grids that exist on *levels*.
- Each level contains a collection of boxes.
- Each box is made up of cells.
- The *refinement ratio* (or `ref_ratio`) is the number of cells on level `l` that fit in a cell on level `l-1` in any one dimension (hardcoded to 2 in GRChombo).
- The physical domain covered by the boxes on level `l+1` are a strict subset of that covered by the boxes on level `l` and there is a buffer of at least 1 (actually forced to be `ceil(num_ghosts/ref_ratio) + 3 = 5` in GRChombo) on level `l` between level `l+1` and level `l-1` (*properly nested (i)*).
- A cell is either fully refined or not refined (*properly nested (ii)*).
- The boxes on every level have a maximum and minimum length in each dimension given by the parameters `max_box_size` (or `max_grid_size`) and `min_box_size` (or `block_factor`).


A hierarchy of block-structured grids in 2D

Regridding

- Suppose we are refining on level `l` where `l=0` is the coarsest level and `l = max_level` is the finest level.
- First we "tag" all cells which require refinement using a *tagging criterion* (see later).
- Now we need to efficiently partition the tagged cells into boxes.
- [GR]Chombo uses the *Berger-Rigoutsos grid generation algorithm* to do this (see Katy's PhD thesis for more details).
- The efficiency of a partition is given by the ratio of tagged cells to total cells in the partition and its minimum is controlled by `fill_ratio < 1.0`.



Is the box:

✓ Properly nested?
✓ Bigger than min box size?
✓ Smaller than max box size?
✓ Sufficiently "full" (above specified fill factor)?

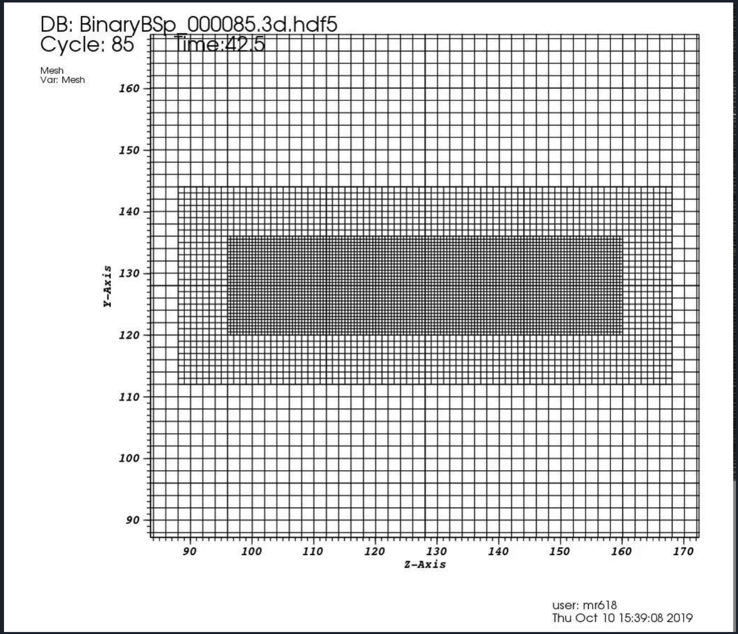
If so then keep and refine, otherwise continue to divide. Stop if box is:

✓ Smaller than min box size

Figure illustrating the process of partitioning grids for refinement based on tagged cells (Credit: Katy's PhD Thesis)

When to regrid?

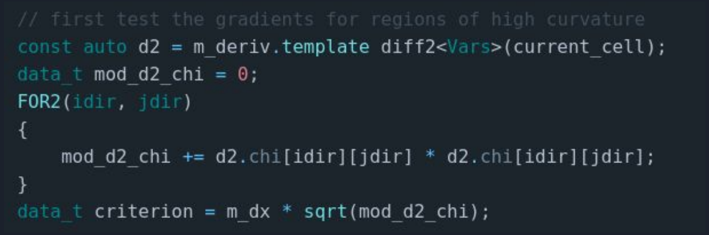
- The number of timesteps between regridding on each level is controlled by `regrid_interval`.
- A regrid on level `l` forces a regrid on all finer levels `>l` so you [probably] don't need to ever regrid on the finest `m` levels for some `m`.
- Regridding too frequently introduces noise that will pollute your solutions.
- Regridding not frequently enough may mean your simulation is underresolved in some regions.



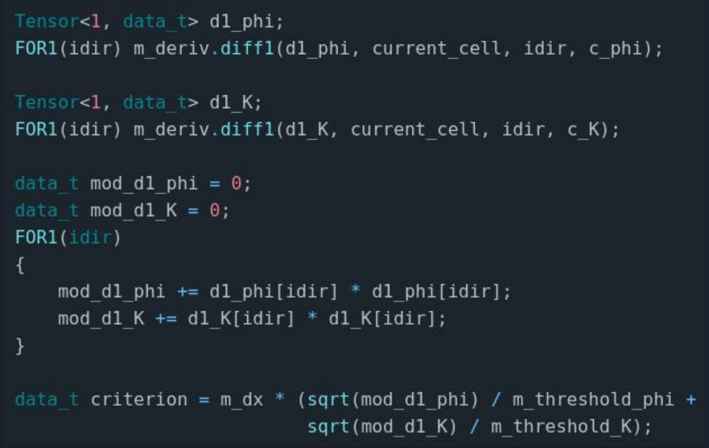
When regridding goes wrong...

Tagging Criteria

- This is a function of the variables in a cell and neighbouring cells (for derivatives) that outputs a real number, `criterion`, in each cell .
- A cell is tagged if `criterion > regrid_threshold` (defaulted to 0.5).
- In some sense completely arbitrary.
- Ideally would like to approximate the truncation error.
- Often, we start by using L^2 norms of undivided differences/second differences (i.e. a derivative stencil not divided by the grid spacing) of well-behaved variables or quotients of them.
- Bear in mind GRChombo adds a buffer of size `tag_buffer_size` (defaulted to 3) around initially tagged cells to the set of tagged cells.



```
// first test the gradients for regions of high curvature
const auto d2 = m_deriv.template diff2<Vars>(current_cell);
data_t mod_d2_chi = 0;
FOR2(idir, jdir)
{
    mod_d2_chi += d2.chi[idir][jdir] * d2.chi[idir][jdir];
}
data_t criterion = m_dx * sqrt(mod_d2_chi);
```

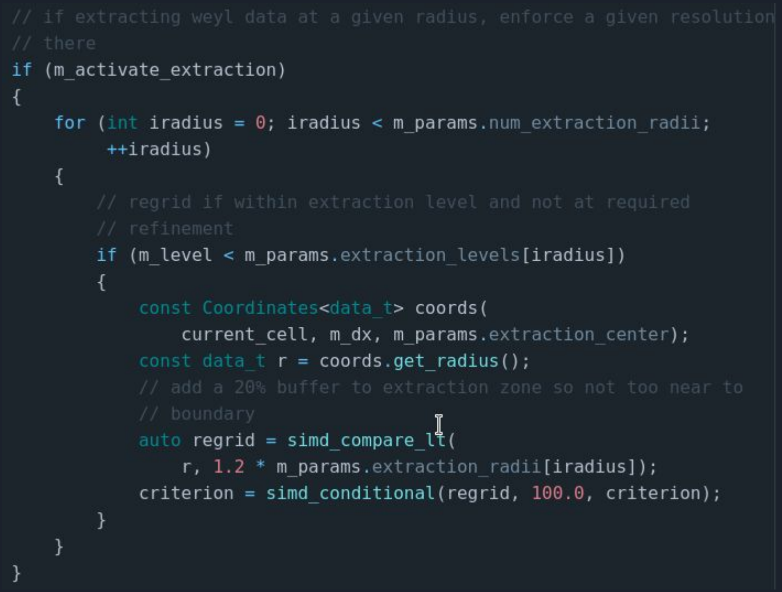


```
Tensor<1, data_t> d1_phi;
FOR1(idir) m_deriv.diff1(d1_phi, current_cell, idir, c_phi);

Tensor<1, data_t> d1_K;
FOR1(idir) m_deriv.diff1(d1_K, current_cell, idir, c_K);

data_t mod_d1_phi = 0;
data_t mod_d1_K = 0;
FOR1(idir)
{
    mod_d1_phi += d1_phi[idir] * d1_phi[idir];
    mod_d1_K += d1_K[idir] * d1_K[idir];
}

data_t criterion = m_dx * (sqrt(mod_d1_phi) / m_threshold_phi +
                           sqrt(mod_d1_K) / m_threshold_K);
```

Some examples of parts of tagging criteria code in GRChombo

Tagging Criteria Tips

- Can use other information to augment the tagging criterion:
  - Apparent Horizon location
  - GW extraction spheres
  - Anything really!
- In general you want your regions of interesting physics as far away from refinement boundaries as possible.
  - Spurious reflections at refinement boundaries add noise.
  - Noise is bad.
  - Black holes seem to be **very** sensitive to this.
- Over to you for your tips



```
// if extracting weyl data at a given radius, enforce a given resolution
// there
if (m_activate_extraction)
{
    for (int iradius = 0; iradius < m_params.num_extraction_radii;
         ++iradius)
    {
        // regrid if within extraction level and not at required
        // refinement
        if (m_level < m_params.extraction_levels[iradius])
        {
            const Coordinates<data_t> coords(
                current_cell, m_dx, m_params.extraction_center);
            const data_t r = coords.get_radius();
            // add a 20% buffer to extraction zone so not too near to
            // boundary
            auto regrid = simd_compare_lt(
                r, 1.2 * m_params.extraction_radii[iradius]);
            criterion = simd_conditional(regrid, 100.0, criterion);
        }
    }
}
```

Some more GRChombo tagging criterion code which shows using the location of the extraction spheres to tag cells.

Any questions?