Explaining “Coarse to Fine” (C2F) feature in Campfire

Fengwei YANG, 6 Jan 2014 *(School of Computing, University of Leeds)*

“Coarse to Fine” (C2F) is a newly implemented feature in Campfire that enables reading in a check-point file from coarser grids, then interpolating data to finer ones and continuing the computation on these finer grids. The motivation behind this implementation is there are applications which steady states are interested, computations during the transition are necessary but can be lengthy if using finer grids. One way to improve the efficiency is to obtain a steady state solution from coarser grids computation then interpolate to finer grids as an improved initial condition. Thus the computation toward steady state on finer grids, will be significantly shortened, but hopefully no less accurate.

The concept of this implementation is simple, given the structure of Campfire and its library PARAMESH. If C2F feature is desired during the run, then we need to initiate the settings for the finest grid that is required at the very beginning of the run. This is to mainly deal with memory allocation. Then during the run, we only need to alter the global parameter $lrefine\_max$ to say which finest grid we want at when.

There were issues during the implementation. The first one was Campfire requires at least 1 block on a grid level in order to successfully initiate. However, when we read in a check-point file from coarser grids, it’s obvious there are no blocks at all on our desired finer grid levels, thus in subroutine $get\_input\_parameters$, where we read in check-point file, we insert a subroutine call to $amr\_relocate\_meshing$, which change no date but will initiate block structure and memory allocation. Another issue was when read in 3D data, it seemed we need to call refine-derefine subroutine or otherwise the program will fail. Therefore, a line is added to $amr\_mg\_time\_dep\_control$ to enable the code to call in refine-derefine subroutine after we read in 3D data.

User-edit is required for a few number of variables in the code in order to get C2F work properly. The first one is $C2F\_desired\_level$, which should be the finest grid level needed, not just the finest grid level when read in. The second one is $C2F\_start\_level$, which is the finest grid level the check-point file has. The third ones are an small array of variables in $amr\_mg\_time\_dep\_control$, they are called $C2F\_turning\_point(i)$. Since multiple jumps are doable in C2F feature, therefore the code needs to know at which time-step we should expand our grid level further. Thus, user is requested to input such information into $C2F\_turning\_point$ array.

It is recommended to generate coarser grids steady state solution from runs that without enabling C2F feature, the reason is C2F allocates memory for finer grids, even though you don’t use these, they still can slow down the code.

Here we list the files and subroutines that had been modified for C2F feature.

- in $amr\_modules.f90$, subroutines: $multigrid\_parameters$ and $generic\_parameters$;
- in $amr\_control.f90$, subroutines: $amr\_mg\_time\_dep\_control$ and $get\_input\_parameters$;
- in $app\_problems.f90$, subroutine: app_domain_setup.