§14. Development of a Multi-Scale Simulation Code with Adaptive Mesh Refinement and its Application to Conventional Codes

Usui, H., Nagara, A. (Graduate School of System Informatics, Kobe Univ.),
Nunami, M.

Adaptive Mesh Refinement (AMR) technique can provide efficient numerical calculation by adapting fine cells to regions where higher numerical resolution is required. However, it is generally difficult for users to implement the AMR technique in their generic simulation programs which use uniform cells. Meanwhile, to investigate multi-scale phenomena in space plasma environment including plasma kinetic effects, we have been developing a new electromagnetic Particle-In-Cell (PIC) code called PARMER by incorporating the AMR technique [1,2]. In the present study, based on the numerical technique on AMR we adopted in PARMER, we started to develop a computational framework for blocked-structured AMR simulation by which we can easily convert a generic uniform-cell simulation program to the one with the AMR treatment [3].

In the framework development, we decided to adopt the block-structured AMR because of better portability than other AMR-structures. In the block-structured AMR, regions required for the AMR treatment in the simulation domain have a self-similar structure. The self-similar block-structured domains for AMR are managed in a fully threaded tree (FTT) data structure which allows recursive refinement on a block-by-block basis. Each block consists of a domain formed with the fixed number of cells with uniform cell size. A block in a different level of refinement in the FTT structure has different cell size keeping the same number of cells. For instance, in one level higher, the cell size becomes half. In each block, we can incorporate the same uniform-cell simulation program of our interest and independently perform the simulation. Since each block has a common domain with the same number of cells because of self-similarity, what we need to consider is the cell size in each block depending on the refinement level which is given in the FTT structure.

A simple example is shown in Figure 1 in which each block has 4×4 cells and the simulation domain consists of 4×4 base-blocks. We call a block with the coarsest cells the base-block. It should be noted that the number of cells in the refined block is the same as that in base-block because of self-similarity although the cell size becomes half. The number of cells in each block and the number of base-block consisting of the whole simulation domain can be initially set as input parameters.

We can easily modify a generic uniform-cell simulation program into an AMR one by inserting it into our framework. Since the framework can handle the hierarchical relation among the blocks with the FTT structure, what the users basically have to prepare is the outer boundary condition for the entire simulation space, the main routine for calculation in each block, and a criterion for the cell refinement. We need to prepare these three parts in the Fortran language. Each block domain has a buffer region which surrounds the physical domain of the block. Through the buffer region, data of adjacent blocks are exchanged for the boundary calculation with finite difference methods. By these systematic connections among blocks, a simulation in a whole domain is efficiently performed.

In parallelizing the code, we use the domain decomposition method with which we uniformly separate the whole simulation region into subdomains with the number of processes available for the parallel calculation. In parallelizing the block-structured simulation, each divided subdomain consists either of a single block or a group of blocks depending on the memory size of each process. In the domain decomposition method we need to exchange the boundary data of each subdomain between adjacent processes. The data exchange between processes is handled with MPI and this treatment is also supported in the current block-AMR framework. From this point of view, the developed framework is also useful for users to parallelize a code with the domain decomposition method using multiple processes even if there is no need to use the AMR function in the simulation.

By taking a two-dimensional advection equation as an example, we performed a test simulation of a square-shaped waveform propagation by using the developed AMR framework. In the waveform propagation, we could confirm that high resolution is achieved adaptively and locally at the steep gradient of the waveform by the creation of new blocks with fine cells. We also confirmed that the computation resources used for this test simulation are reduced almost by half in comparison with those used in a uniform cell simulation for the present case.

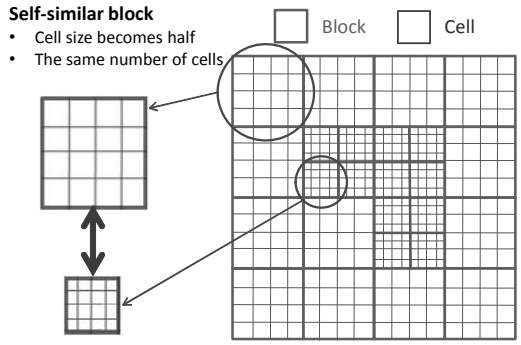

Fig. 1. Example of block configuration in two-dimensional domain. In this case, a block has 4×4 cells.

1) Usui, H. et al.: Procedia Computer Science **4** (2011) 2337.
2) Usui, H. et al.: Plasma and Fusion Research **8** (2013) 2401149.
3) Usui, H. et al.: Procedia Computer Science **29C** (2014) 2351.