# Implementation of AMR Technique into Parallelized Plasma Codes and Its Effectiveness

Masanori Nunami [1], Hideaki Miura [1], Akihide Nagara [2], Masaharu Matsumoto [2], Hideyuki Usui [2], and Ryosuke Goto [3]

[1] National Institute for Fusion Science, 322-6 Oroshi-cho, Toki 509-5292, Japan
[2] Graduate School of System Informatics, Kobe University, 1-1 Rokko-dai Nada-ku, Kobe 657-8501, Japan
[3] Graduate University of Advanced Studies, 322-6 Oroshi-cho, Toki 509-5292, Japan

## 1 Introduction

To study a complex dynamical behaviour of plasmas, e.g., space plasmas, laboratory plasmas and fusion plasmas, numerical simulations with high-resolution in space and time are prerequisite subjects. They are required not only to understand detailed physics of small-scale events such as saturations of short wave instabilities, micro turbulence and magnetic reconnections, but also to construct a macro-scale model to predict long-time behaviours of the plasmas because small-scale events often affect the long-time behaviours through changing local pressure gradients and/or magnetic configurations. In such a situation with high-resolution of the calculation grids, we need a huge number of grid points to model the region to guarantee the numerical stability. In the conventional codes which employ uniform grid system, due to the recent accelerative gains of massive computational powers such as “Kei” computer, it has learned to calculate the behaviours with a certain resolution level. Nevertheless, in order to cover whole phenomena from the micro-scale events to the macro-scale behaviours of the plasmas, it is not efficient at all in terms of the usage of limited computer resources such as the amount of memory and calculation time.

For the purpose of carrying out the numerical simulations with high resolution and a high efficiency in the possible calculation time and computer resources, the Adaptive Mesh Refinement (AMR) algorithm is one of the strongest candidates. In the AMR simulation, grids with different spacing are dynamically created in hierarchical structures according to the local conditions of phenomena. Fine grids suitable to the local high density region are applied only there and other regions are simulated by using moderate size grids. Therefore, increment of the numerical cost due to the localized region is not serious by adopting the AMR technique. On the other hand, since the AMR algorithm has to treat hierarchical grid system, the AMR code should be programmed with extremely different structure of the computation flows and data set from the conventional simulation codes with uniform grid systems. In this paper, in order to implement the technique into the conventional plasma simulation codes, e.g., magneto hydrodynamic (MHD) code, kinetic plasma code, and so on, we developed the AMR module that is based on block-structured AMR framework. The effectiveness of the parallel computation in several computer architectures is also discussed.

## 2 Block-Structured AMR Framework

AMR techniques were implemented by several schemes, for example, cell-based refinement structure, patch-based meshes, block-structured framework, and so on. Furthermore, the techniques were applied for many scientific fields. Cell-based scheme is advantageous for saving the computer resources compared with other schemes, since we can save the refinement regions with minimum requirement resources. An AMR code, PARMER [1] is plasma particle code with AMR technique by using cell-based refinement structure and it has good efficiency in terms of saving the computer memories [2]. However, the cell-based implementation should uses quite different structure of data and grid systems from the conventional code that uses adjustable arrays. Therefore, for more good portability of implementation of the AMR method, the block-based AMR framework is more advantageous than the cell-based schemes. The block-structured AMR method was originally developed by Berger and Oliger [3] for hyperbolic equations. In this approach, self-similar blocks divide whole space of the simulation system. Since each block has self-similar grid systems with the size ($N_x$,$N_y$,$N_z$), several conventional simulation codes based on adjustable array systems are applicable in each block. As shown in Fig. 1, the developed module [4] employs the block-structured AMR system and fully-tree-treaded (FTT) data structure [4] with three types of pointers, ‘neighbour’ for the

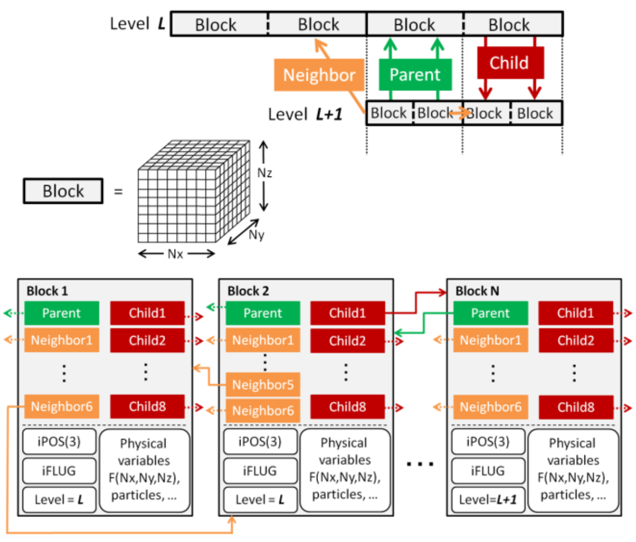

Fig.1 A schematic illustration of the data structure adopted in the developed AMR module. Arrows indicate the pointers. Three types of pointers, neighbour, parent, and child, are used to specify the relationship between two Blocks. For simplicity, a case of two hierarchical levels is depicted in the top panel. The bottom panel shows the detail of the Block structure. In each block, self-similar rectangular grid systems with the size ($N_x$,$N_y$,$N_z$), physical variables such as fields, particles, and so on, and some flags for the refinement procedure are contained.

surrounding grids, ‘parent’ for the original coarse grids in the lower hierarchy, and ‘child’ for the refined grids in the higher hierarchy. These pointers are included into a *structure* of Fortran 90, and used to specify the relationship between two blocks. By using the FTT data structure, it becomes easy to handle the creation or removal of each hierarchical domain during a simulation run.

## 3 Parallelization and Effectiveness of the AMR module in Plasma Simulations

In various plasma simulation studies with particle codes, hydrodynamic codes, and so on, domain decomposition method is one of most familiar parallelization schemes. For the developed AMR module, we adopt the domain decomposition scheme for the parallelization using MPI library. While the AMR module is based on the block-based approach and the self-similar refinement, it is designed to be applicable to the parallelization easily, because the communication between the blocks is quite similar to that between the decomposed calculation domain nodes. Thus, in the module, the communications for parallel processing should be executed at boundaries of each block.

To demonstrate the effectiveness of the developed AMR module in parallelized plasma simulations, we carry out MHD simulations. The MHD equations are approximated either by the use of the 4th order central finite difference scheme or by the use of the 8th order compact finite difference scheme, which have been applied for the nonlinear MHD simulations of the Large Helical Device [6,7]. The suitability of the compact finite difference scheme to the AMR approach is also discussed. Since the block structure variable can include particles easily in each block domain, the developed module is applicable to plasma particle simulations. The portability of the module is another issue in the development. In this work, we demonstrate the plasma simulations with the AMR module in several computer architectures, e.g., HITACHI SR16000, Bull B510 and Cray XE6.

## 4 Summary and Discussions

We have been developing an AMR module for parallelized plasma simulation codes, and demonstrating the effectiveness of the module in the several computers. The developed module is designed for good portability into conventional plasma simulation codes and any computer architectures. In this work, we demonstrated the effectiveness of the module in different computers and simulation codes, i.e., MHD code and particle code in plasma physics.

The open difficulty in the development of the AMR module is the load balancing of the parallelization between processors, because the number of spatial grids or particles belonging to each sub-domain is not always constant due to the AMR procedure. In each sub-domain, the hierarchical grid layers are adaptively created or deleted. To achieve the load balancing among sub-domains distributed to processors, we already obtain the method based on decomposition by the Morton ordered space-filling curve [8] in the development of PARMER code. With the help of the technique of PARMER, we should number all the blocks with the Morton ordered space-filling curve and divide the order into the number of the processors in such a manner that each sub-domain has approximately the same amount of the calculation costs, for example, numbers of blocks, particle calculation loops, and so on.

## Acknowledgements

This work was supported in part by JST (Japan Science and Technology Agency)/CREST (Core Research for Evolutional Science and Technology), and in part by Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant No. 23340182. The computations in the present study were performed with Plasma Simulator of NIFS, Helios of IFERC-CSC, and KDK of RISH at Kyoto University.

## References

[1] H. Usui, M. Nunami, T. Moritaka, et al., *A Multi-Scale Electromagnetic Particle Code with Adaptive Mesh*, Procedia Computer Science **4**, 2337 (2011).

[2] H. Usui, Y. Yagi, T. Matsui, et al., *Dynamic Domain Decomposition for 3D PIC simulation with Adaptive Mesh Refinement*, 21st International Toki Conference, Toki, Japan, P1-79 (2011).

[3] M. Berger, J. Oliger, *Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations*, J. Comp. Phys. **53**, 484 (1984).

[4] H. Miura, M. Nunami, H. Usui, et al., *Development of a portable AMR module applicable to various fluid/particle simulations*, 21st International Toki Conference, Toki, Japan, P1-77 (2011).

[5] A. M. Khokhlov, *Fully Threaded Tree Algorithms for Adaptive Refinement Fluid Dynamics Simulations*, J. Comp. Phys. **143**, 519 (1998).

[6] H. Miura, N. Nakajima, T. Hayashi et al., *Nonlinear Evolution of Magnetohydrodynamic Instability in LHD*, Fusion Science and Technology **51**, 8 (2007).

[7] H. Miura, N. Nakajima, *Influences of ballooning modes with moderate wave number on MHD equilibrium in LHD*, Nuclear Fusion **50**, 054006 (2010).

[8] M. S. Warren, J. K. Salmon, Proceedings of Supercomputing, IEEE **12**, 21 (1993).