Development of a Block-Structured AMR Module

A. Nagara [1], M. Nunami [2], M. Matsumoto [1],and H. usui [1]

[1] Graduate School of System Informatics, Kobe University, Japan
[2] National Instutite for Fusion Science, Japan

1 Introduction

The aim of this work is the development of a high-resolution and high-efficiency module applicable to various numerical simulations. Despite the increasing computational power, lack of numerical resolution is a persistent problem in a simulation studies.

To solve this problem, we adopted adaptive mesh refinement (AMR) which realizes both high-resolution and high-efficiency in a simulation [1,2]. AMR technique can provide efficient numerical calculation by adapting grids to regions where higher numerical resolution is required. However, it is difficult to implement the AMR technique in a conventional simulation code. The development of a portable AMR framework will contribute to the high performance of various numerical simulations.

2 Design of AMR module

We adopted block-structured AMR [2,4], in consideration of portability. The module has two point for portability.

2.1 AMR-type

The module is based on self-similar block-structured AMR, with fully threaded tree data structure (FTT) allowing recursive grid refinements on a block-by-block basis[5]. In each block, uniform grid code is applicable because of block-structure, and it is possible to calculate with same size grid system for each refinement levels because of self-similar structure. (Fig.1)

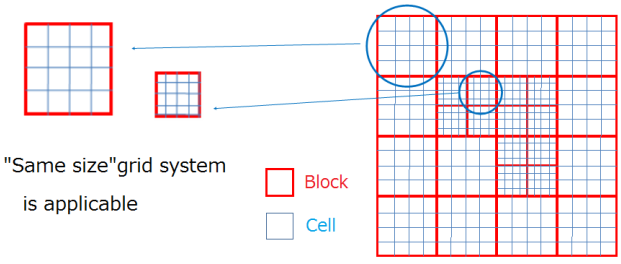

Fig.1 Self-similar block-structure AMR

Each block has information about level of refined hierarchy and physical variable. Every block has three kinds of pointers which imply neighbor, child, and parent blocks. Relationship of each block is built by pointers. Each block is treated as independent unit organized in refinement tree.

2.2 Proccessing flow

The module is designed to isolate calculation in a block from AMR process. Because of the isolation, application of a uniform grid code needs to change only the calculation in each block. (Fig.2)

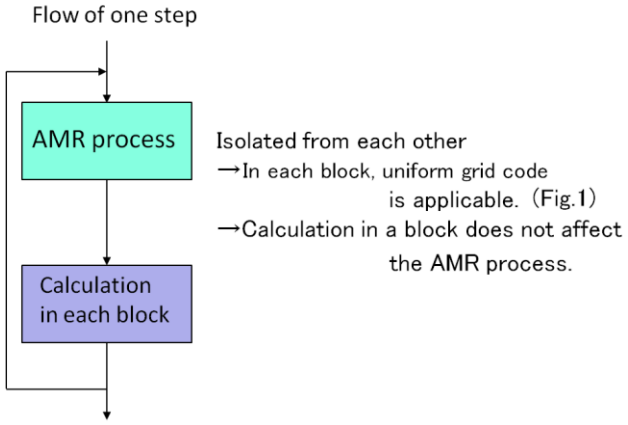

Fig.2 Processing flow of one step calculation

Due to the isolation, processing flow contains the overhead. Calculating the internal arrays of each block, the module needs the three processes. First, in each block, the contents of the internal arrays are copied to temporary arrays for smooth calculation. Second, by calling the uniform grid code which is introduced to the module, physical values in temporary arrays are calculated. Finally, the updated values in the temporary arrays are copied to the internal arrays of the original code. At the cost of partial efficiency, the module pursues the portable framework.

3 Example of AMR module incorporation

To demonstrate the effectiveness of the module, we applied the module to simple 1D/2D advection equation.

3.1 High-resolution by AMR module

In this test case, the module creates one subdivision block. With the introduction of this module, advection equation should be calculated with double accuracy. High-resolution advection calculation is confirmed by comparison between cases of AMR and uniformed grid (Fig.3). Figure 3 shows calculation results of 1D upwind-difference-scheme(UDS) calculated by the AMR module and uniform grid code. The vertical axis shows the physical quantity, and the horizontal axis shows the position in fig.3. This graph indicates that regions where physical quantities has large gradient are refined by AMR technique. It was shown that using the AMR module can provide high resolution calculation.

3.2 High-efficiency by AMR module

Measurement of computing time shows that the application is high-efficiency module compared to uniform grid codes (Fig.4). Figure 4 shows the comparison of computing time for 2D UDS calculated by the AMR module and uniformed grid code. The vertical axis shows the conputing time, and the horizontal axis shows the size of simulation field in fig.4. In all cases , the

initial condition is common and only the size of simulation field is different. Physical quantities were calcurated with the same accuracy. The increase rates of computing time suggest that calculation of 2D UDS using the AMR module is faster than calculation by uniform grid code. When the simulation region become large, the result by the module is become efficient.

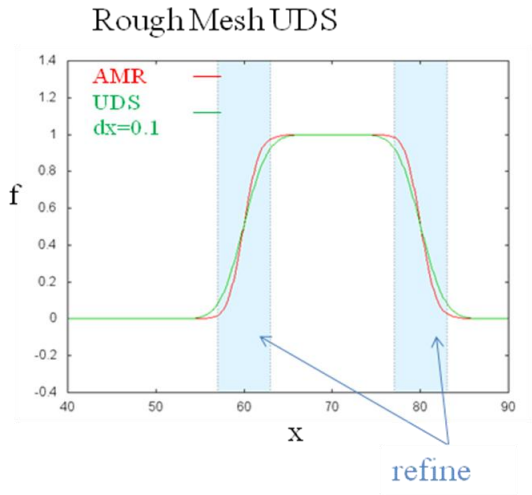

Fig.3 Comparison of 1D AMR and 1D uniform grid code

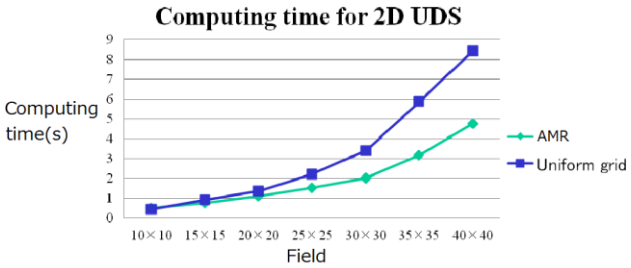

Fig.4 Computing time for 2D UDS

Comparison of memory usage has a same tendency of comparison of calculating time; calculation by the module needs lower memory usage than calculation by uniform grid code when the simulation region is large. The module makes for efficiency of calculation. In the case of small simulation region, however, computing time and memory usage using the module are about the same as these using uniform grid codes.

In order to calculate efficiently even if the calculation region is small, it is necessary to reduce the overhead of blocking. We should explore more efficient method of blocking without sacrificing the portability.

4 Conclusion

The AMR module developed in this work is effective to improve the accuracy , and allows for the efficient computation to some extent. Considering the overhead of the module, more efficient blocking method depending on introduced calculation is required. Even when simulation size is small, we should make the module calculate more efficient.

In terms of portability, we have introduced only a few calculation examples, so it is necessary to try to introduce the various calculations. For the future, the module will be used in several simulation code, MHD code, Gyrokinetic Vlasov code, and Particle code. Evaluation of portability will be appreciated soon after.

The module has function of only one step subdivision and does not adapted to parallelization. In order to apply to large-scale calculations, we consider that the contents of the following development are parallelization, n-step subdivision and more efficiency blocking technique.

References

[1] Darren De Zeeuw and Kenneth G.Powell, "An adaptively refined Cartesian mesh solver for the euler equations." *J.Comput.Phys*. 104:56-68,1993

[2] J. J. Berger and J. Oliger. "Adaptive mesh refinement for hyperbolic partial differential equations." *J.Comput.Phys.* 53:484-512,1984.

[3] A. V. Kravtsov, A. A. Klypin, A. M. Khokhlov, "Adaptive refinement tree—A new high-resolution N-Body code for cosmological simulations" *Ap. J. Suppl.*, 111 p. 73, 1997

[4] M.L. Norman and G.L. Bryan. "Cosmological adaptive mesh refinement. "*astro-ph.*/9807121, July 1998.

[5] A. M. Khokhlov. "Fully Threaded Tree Algorithms for Adaptive Refinement Fluid Dynamics Simulations." *J.Comput.Phys.* 143:519-543,1998