Applications of parallel computation with sharing information process in Numerical General Relativity and Adaptive Mesh Refinement in Magnetohydrodynamics.

Mariana Cécere[1], Santiago Gómez[2], Luis Lehner[3], Oscar Reula[4]

F@MAF- Universidad Nacional de Córdoba[124]

Department of Physics and Astronomy - Lousiana State University[3]

email: cm4@famaf.unc.edu.ar[1] , santiago.miguel.gomez@gmail.com[2]

keywords: relativity, parallel coding, 3-sphere, adaptive mesh refinement

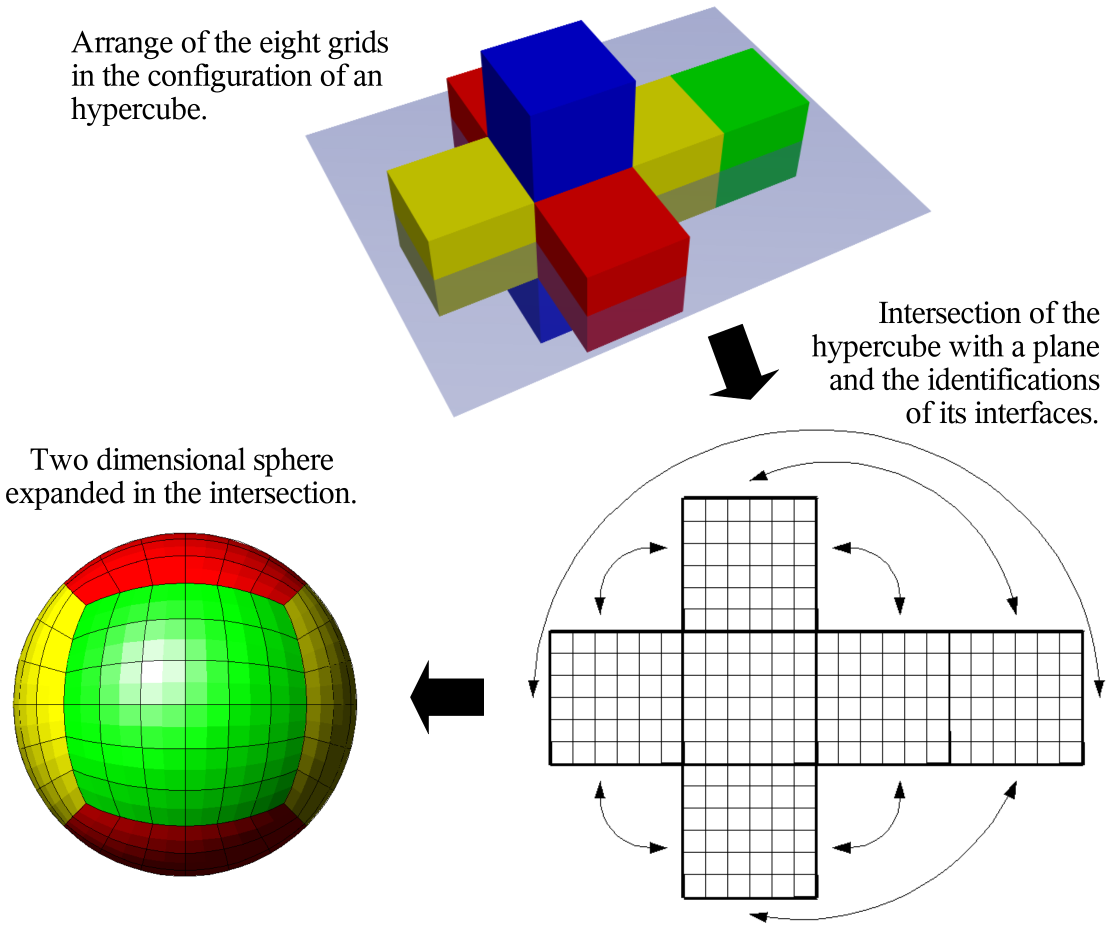
Abstract

Except for very particular cases, most solutions of Einstein's equations are not known in closed form, so numerical simulations are required which in turn require employing high performance computing to accurately represent relevant systems. The research of the world wide community in this field is geared into implementing numerical simulations. The Adaptive Mesh Refinement (AMR) method and the parallel coding with sharing information using MPI (Message Passing Information) routines has become a very efficient technique for realizing these simulations [1]. In the present work, it's developed a code which shares information on a globally closed and periodic discretization grid of a three dimensional sphere. The outgoing information coming only from the boundary of the numerical grid of each parallel process has to be shared with the other processes at each time step during the evolution. For more advanced instances of the research in Numerical Relativity, it comes necessary to incorporate several essentially different phenomenas coexisting in a single numerical evolution, for example gravity and magnetohydrodynamics. Those phenomenas generally evolve at different temporal scales and are localized at different spacial domains. The optimization of the computation for the most relevant part of those phenomenas is performed using the AMR method. It begins by defining a coarse mesh that covers the entire computational domain. Refined grids of higher resolution are added to regions of the domain where additional resolution is required. This process of adding finer and finer meshes continues to some prespecified level of accuracy. In addition, this hierarchical structure is dynamic so that the algorithms are capable of adapting themselves to arbitrary problems by automatically refining and moving meshes to resolve small scale features as they develop and evolve. The result is a tremendous savings of computer memory and a reduction in execution time over large fine grid simulations.

Parallel computation with sharing information process on the 3-sphere:

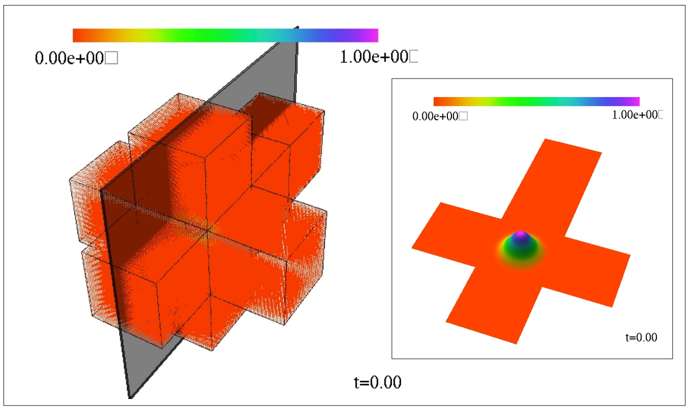
The global process is composed by eight copies of independent parallel processes, where each one performs the computations on its own specific three dimensional grid. The eight grids of the eight parallel processes actually form the three dimensional sphere as it's showed in the next figure. Each grid is a cube, a three dimensional array of $(N+1)^3$ points. The global process only needs to share the information between points at the faces of those cubes. There are $6(N+1)^2$ boundary points per grid and each grid shares information only with six neighbor grids.

Figure 1. Discretization of a three dimensional sphere.

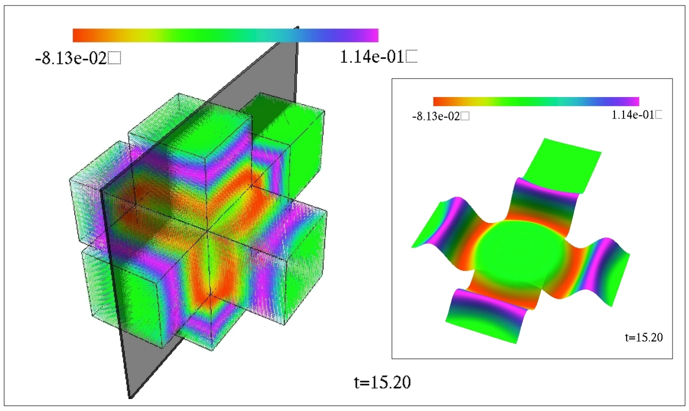

Arrange of the eight grids in the configuration of an hypercube.
Intersection of the hypercube with a plane and the identifications of its interfaces.
Two dimensional sphere expanded in the intersection.

The computation inside each grid is basically implemented by finite differences method with fourth order accurate derivative operators, where the global time evolution is carried out by a fourth order Runge-Kutta scheme. In that way this code can solve any time dependent problem in physics which is reduced to a system of first order partial differential equations on geometry of the three dimensional sphere. The main computational cost actually relies in the fact that large amounts of memory are necessary in order to store four temporal steps data values for each variable in the grid, the available memory for each process fixes the maximum grid resolution. Furthermore, to have a consistent, stable and converging finite differences method scheme a penalty method [2] is applied to the incoming modes of the equations, namely the incoming information from the neighbor cubes. Also one needs to have smaller temporal step evolution with higher grid resolution, so if the resolution is doubled then the total number of temporal steps must also be doubled increasing even more the whole computing time. As an example of a simple time dependent problem which can be solved with this technique, the next figure shows two instants obtained from the simulation, using the described code, of an initial pulse of a massless Klein-Gordon scalar field on a fixed background metric with the S^3 topology.

Figure 2. Sequence of two instants of the evolution of a scalar field on the 3-sphere.



Initial pulse localized in a small area inside one grid.



The initial pulse has propagated trough the interfaces and get inside the neighbor grids.

Each grid has a resolution of N=40, and the first order PDE system in this case is composed of five variables, namely the scalar field and its temporal and spacial derivatives. The initial data pulse has compact support inside a small area inside one specific grid. It can be observed that the scalar field leaves this initial grid which is placed in the center and propagates into its neighbor during the evolution. At each time step of the evolution the MPI routine send and receive the information of the outgoing and incoming physical modes of propagation through each interfaces. Although all the grids are identical, each one has its own specific coordinate system, so the derivatives of the scalar field in each process are calculated in terms of its respective coordinate system. Once each process receives that information from the neighbor grids, it must be decoded by means of a predefined coordinate transformation depending on which grid these derivatives are coming from.

Adaptive Mesh Refinement in Magnetohydrodynamics:

We use the HAD infrastructure, a modular code for solving hyperbolic and elliptic differential equations with distributed parallel AMR [3]. HAD uses Berger–Oliger (B&O) [4] style AMR with sub-cycling in time. Refinement criteria may be problem specific, or a shadow hierarchy allows one to easily estimate the truncation error dynamically for use in specifying refinement criteria. We use hyperbolic divergence cleaning to control the $\nabla \cdot \vec{B} = 0$ constraint for the magnetic field. Finally, we use third-order Runge–Kutta scheme to integrate the equations. The code is written in Fortran for simplicity and easy of debugging. Grids contain fields and associated other data. Levels consist of all fields at a given resolution. Hence, level zero consists of just the coarse grid (or all the grids into which the coarse grid may be domain decomposed for distribution to all the processors), while level one consists of all children of that coarse grid.

The code is written to allow for various projects. The code is distributed using Message Passing Interface (MPI) in such a way that grids are sent to the various processors.

Shadow Hierarchy:

A key ingredient to the B&O algorithm is to know where a refinement is needed. This is accomplished by determining an estimate of the error in the solution for each location and time. The simplest solution is by using some function of the evolved fields, such as a density or gradient. B&O specify another way, called truncation error estimation (TRE). To get an error estimation of the truncation error, one takes a two time steps on a given grid and compares the obtained values with values found by taking a single step on a grid with half the resolution. This difference is indicative of the truncation error.

Parallelization: Multiple Instruction, Multiple Data (MIMD):

- Multiple Instruction: every processor may be executing a different instruction stream.
- Multiple Data: every processor may be working with a different data stream.
- Execution can be synchronous or asynchronous, deterministic or non-deterministic.

References:

[1] L. Lehner, O. Reula and M. Tiglio, *Multi-block simulations in general relativity: high order discretizations, numerical stability, and applications.* arXiv:gr-qc/0507004.

[2] L. Lehner, D. Neilsen1, O. Reula, and M. Tiglio, *The discrete energy method in numerical relativity: Towards long-term stability*, arXiv:gr-qc/0406116

[3] M. Anderson, E. Hirschmann, S. L. Liebling and D. Neilsen, Class. Quantum Grav. 23, 6503 (2006).

[4] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. J. Comp. Phys., 53, 484, (1984).