Mem. S.A.It. Suppl. Vol. 4, 55

Memorie della
Supplementi

GAMMS: an AMR multigrid gravity solver code

C. Gheller[1], F. Sartoretto[2], M. Guidolin[2]

[1] CINECA, via Magnanelli 6/3, Casalecchio di Reno, Bologna, Italy
[2] Università di Venezia, Dipartimento di Informatica, Via Torino 155, Mestre VE (Italy)

Abstract. We describe our GAMMS (Gravitational Adaptive Mesh Multigrid Solver) code, which is a numerical gravitational solver, based on a multigrid algorithm and a local refining mesh strategy (AMR, Adaptive Mesh Refinement). The multigrid algorithm allows for an efficient and accurate evaluation of the gravitational field, while AMR provides high resolution computations exactly where needed. GAMMS is an open source code which takes advantage of the open source DAGH parallel library, based upon MPI. Our code can be integrated with both N-body and fluid-dynamics codes, in order to perform high resolution astrophysical simulations.

1. Introduction

The gravitational forces drive the behavior and the evolution of physical systems in many astrophysical problems. Accuracy in the estimation of these forces is critical for the description of a large number of astrophysical phenomena.

To solve these problems, usually a large computational domain is to be treated, while good spatial resolution is required. Cosmological problems are distinguished examples. A cosmological structure formation process involves an extremely large dynamical range. As large as the universe volumes are featured (10^3 Mpc), yet one must solve the problem of resolving star formation regions in galaxies, which are typically at the 1 pc scale. The spatial dynamical range per dimension required is as large as 10^9, well beyond the maximum value, 10^3, that nowadays can be managed on supercomputers.

A solution to this range problem relies upon combining the Multigrid (MG) method (Brandt , 1977), (Suisalu & Saar , 1995) with local refinement techniques. Multigrid solves differential equations using a set of multilevel grids, thus providing an *optimal* $O(N)$ computational cost, on an N–node grid. Moreover, MG allows for efficiently tracking errors, and it can be effectively combined with local refinement techniques. We exploited Adaptive Mesh Refinement (AMR) (Berger & Oliger , 1984), which allows for dynamically defining and managing the computational meshes; they can be either refined, where high resolution is required, or coarsened, where low resolution is enough. Hence, the treatment of any interesting physical process is performed at its proper spatial resolution, whereas it would be too expensive if an uniform grid treatment is exploited.

The ensuing code, GAMMS (Gravitational Adaptive Mesh Multigrid Solver) is described in the sequel.

2. Multigrid Algorithm

Let us consider Poisson's equation

$$\Delta u(\mathbf{x}) \;=\; \mathbf{F}(\mathbf{x}), \quad \mathbf{x} \in \mathbf{\Omega},$$

where $u(\mathbf{x})$ is the gravitational potential, $F(\mathbf{x})$ is the mass density field, Δ is the Laplacean operator and Ω is the problem domain. Our Multigrid technique estimates the solution u by a Finite Difference (FD) scheme on a uniform, square grid (base grid) on Ω, thus yielding a linear system of equations. The linear system is solved using a relaxation method. The solution u is approximated by a Grid Function (GF), i.e. by assigning an approximated value to each mesh node.

As for the relaxation scheme, we used the Gauss–Seidel *Red–Black* (RB) relaxation scheme (Demmel , 1997), which has appealing parallel features. It consists of a sequence of integration sweeps, ending as soon as the required degree of accuracy is reached. The $(n+1)$–th approximation to $u(\mathbf{x})$ on cell (i, j, k), $u^{(n+1)}_{i,j,k}$, is computed using the standard, central finite difference approximation of the Poisson equation:

$$u^{(n+1)}_{i,j,k} = \frac{1}{6}(\bar{u}_{i+1,j,k} + \bar{u}_{i-1,j,k} + \bar{u}_{i,j+1,k}+$$

$$+\bar{u}_{i,j-1,k} + \bar{u}_{i,j,k+1} + \bar{u}_{i+1,j,k-1}) - \frac{h^2}{6}F_{i,j,k} \ ,$$

where $\bar{u}_{i,j,k}$ is computed as follows. In order to improve the convergence rate of the scheme all the values pertaining to "red" nodes, i.e. such that $(i+j+k)$ is even, are computed first. Hence at this stage for all "black" nodes, i.e. $(i+j+k)$ is odd, we set

$$\bar{u}_{i,j,k} = u^{(n)}_{i,j,k}.$$

After that, the values on "black" nodes are computed using the updated "red" values: when $(i + j + k)$ is even, we have

$$\bar{u}_{i,j,k} = u^{(n+1)}_{i,j,k}.$$

Relaxation is a local algorithm, since the solution on a mesh node is based upon only the adjacent nodes. Therefore it can efficiently reduce the error only at high frequency modes, while performing bad on smooth error components, which live at a *whole domain* scale. On the other hand, large scale smooth errors can be read as local errors on coarser grids, where they can be smoothed out using the same relaxation technique. Several discretization levels of the same continuous problem can be used to smooth out errors at different scales. This is the basic idea of MG, which enrolls a hierarchy of coarser and coarser grids to reduce the error at larger and larger scales, thus providing fast convergence.

To go up and down along the grid hierarchy, two problem–dependent operators must be devised: a *restriction operator*, which allows to map any GF given on a fine grid, onto a coarser grid; a *prolongation operator*, in order to map any GF given on a coarse grid, onto a one–level higher, finer grid.

Traversing the hierarchy is called performing a *V-cycle*. Figure 1 sketches a sample V-cycle.

Since coarsening produce smaller algebraic systems, the computational cost of solving a linear system on a coarse grid is low. One can prove that MG has an $O(N)$ computational cost, N being the number of grid nodes.

MG can also easily manage locally refined grids.

3. AMR Technique

The AMR technique (Berger & Oliger , 1984) allows to refine any region in the computational volume by building, at *each grid level*, l, a nonuniform grid, which is the union of uniform sub-grids G^l_0,...,G^l_M, each one featuring its mesh size $d_k = d_{k-1}/2$, $k = 1, \ldots, M$; d_0 being the diameter of G^l_0. Each sub-grid does not cover the entire domain. The region covered by G_{k+1} is only a part of that one encompassed by G_k (but not vice versa). Therefore, we have many refinement levels on each computational sub–domain (see Figure 2).

The multigrid algorithm can easily manage nonuniform grids, by exploiting the same restriction and prolongation operators which allows for wandering through the grid level hierarchy.

Inside our GAMMS code, each adaptive mesh is built and controlled using the DAGH (Distributed Adaptive Grid Hierarchy) library (see `http://www.caip.rutgers.edu/~parashar/DAGH`). DAGH provides data

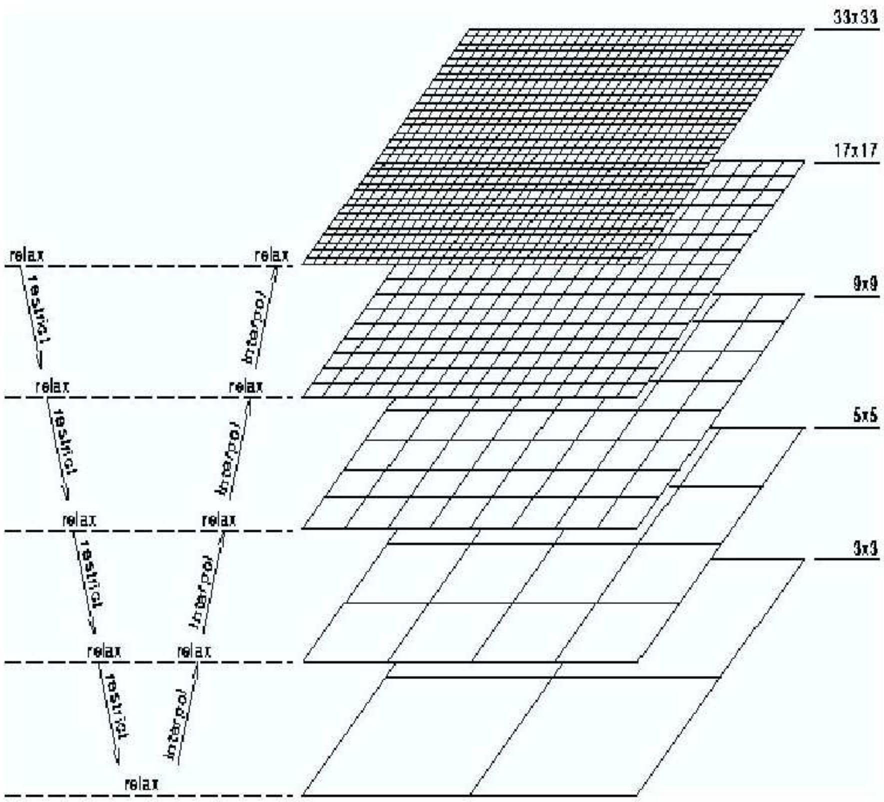

Fig. 1. Sketch of an MG V-cycle on a grid hierarchy.

abstractions and all the tools required in order to define and manage grid values on a nonuniform, dynamically evolving mesh (figure 3). DAGH is an open source package and features a parallel implementation, based upon a standard MPI library. The algorithm for local refining the computational mesh is based on the AMR scheme described in (Berger & Colella , 1989), which appropriately defines a sequence of sub-domains that do not describe the whole domain Ω, but only the regions where higher resolution is needed. These regions can be identified either using an error based criterion (which comes together with the multigrid technique), or using the matter density field, locating high resolution sub-domains where highest peaks in the density distribution appear. Once the subdomains are identified, they are defined as rectangular regions *properly nested*, i.e.

1) grids at the same level do not overlap;
2) a fine grid starts and ends on the corner of a cell in the next coarser grid;

Hence, a fine grid has to be included into one and only one coarser level grid (see Figure 2). The grid hierarchy can be recomputed at any stage of a simulation, according to the behavior of the analyzed physical system

4. Results

At the present stage, MG and AMR are not fully combined. We tested them separately. Tests on our MG implementation proved that the algorithm converges in few relaxation sweeps, when a moderate number of V-cycles is performed (figure 4). We performed some

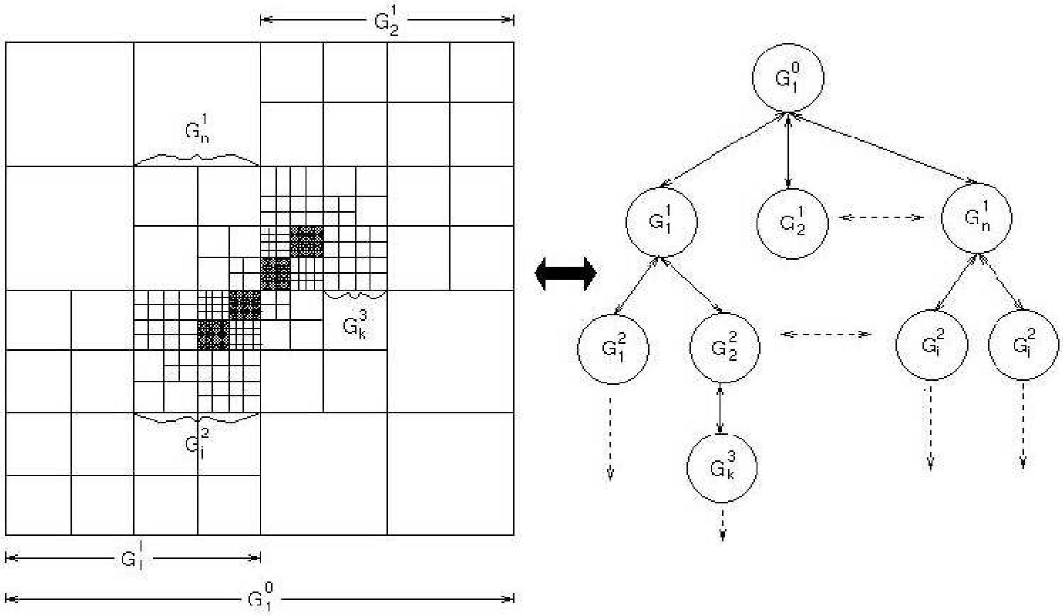

Fig. 2. A locally refined grid and related data structure.

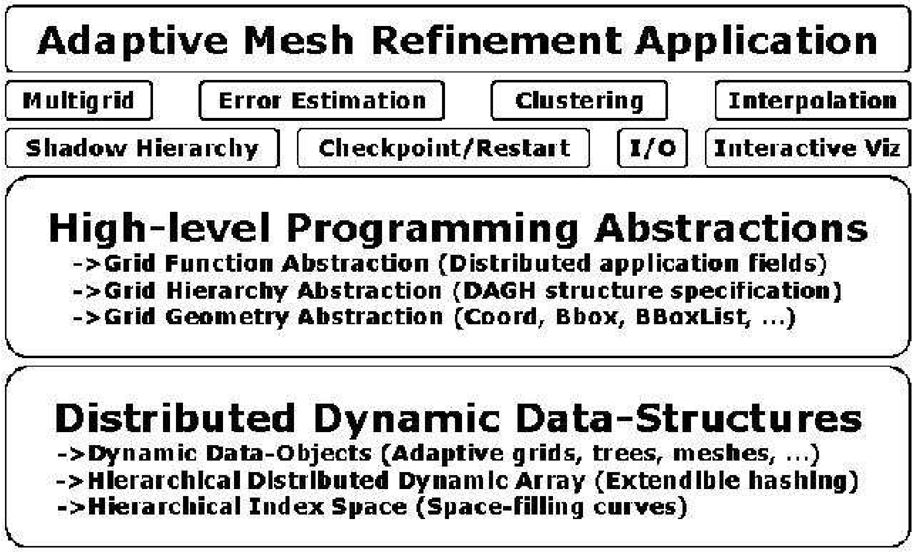

Fig. 3. Structure of a DAGH based AMR application

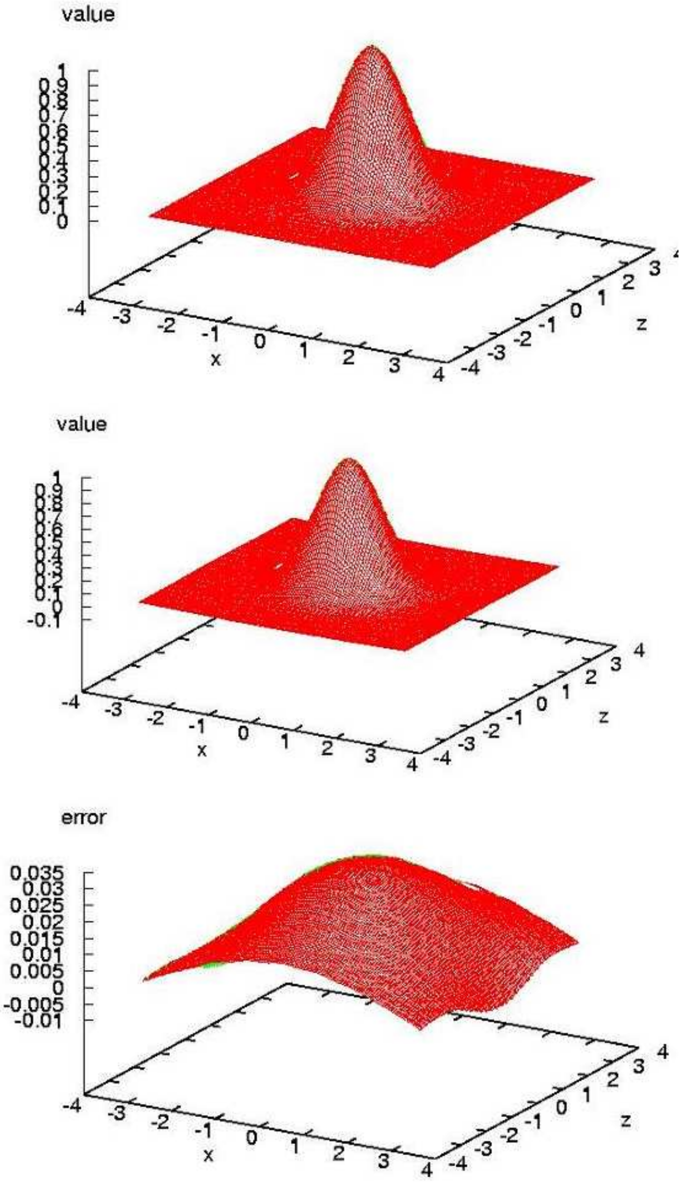

Fig. 4. Comparison between calculated and analytical solutions and relative error for a gaussian shaped gravitational potential.

test runs on the IBM SP4 located at CINECA (Italy). Figure 5 shows that the computational cost of our MG code increases linearly with the number of grid points, as theory predicts. Tests on our AMR code show that it can efficiently handle an adaptive mesh. Figure 6 shows a three–level local refinement driven by a Gaussian function, which was initially set on the coarsest, uniform grid. The grid was locally refined according to the magnitude of the derivative: larger magnitude requires finer resolution.

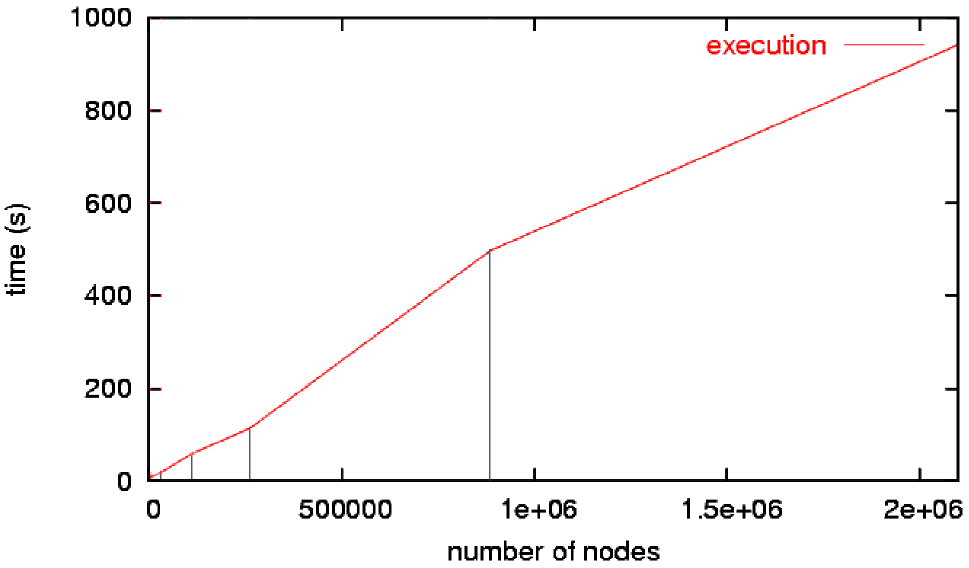

Fig. 5. CPU seconds spent by MG vs number of grid points.

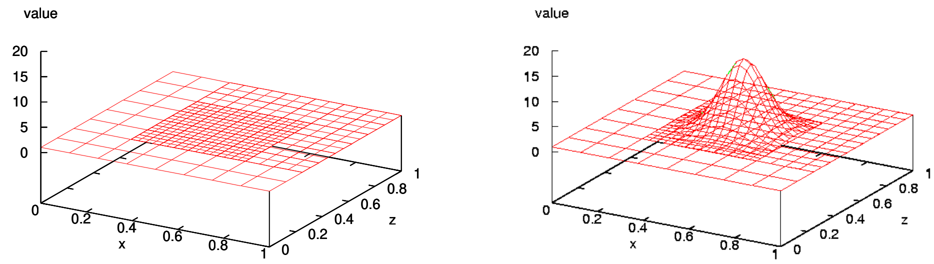

Fig. 6. Hierarchical grid structure adapted to a Gaussian function. Left: the locally refined mesh. Right: a Gaussian function calculated on the refined mesh.

References

Berger M.J., Colella P., 1989, J. Comput. Phys., 82, 64
Brandt A., 1977, Math. Comp., 31, 333
Berger M. J., Oliger J., 1984, J. Comp. Phys., 484, 54
Demmel J. W., 1997, Applied Numerical Linear Algebra, SIAM.
Suisalu I., Saar E., 1995, MNRAS, 274, 287S