Mem. S.A.It. Suppl. Vol. 1, 96

Memorie della
Supplementi

AstroComp: using the portal to perform astrophysical N-body simulations

P. Miocchi[1], V. Antonuccio[2], U. Becciani[2], R. Capuzzo Dolcetta[1], A. Costa[2], P. Di Matteo[1], and V. Rosato[3],

[1] Dipartimento di Fisica, Universitá di Roma "La Sapienza", P.le A. Moro, 2, 00185 Roma, Italy e-mail: miocchi@uniroma1.it

[2] INAF, Osservatorio Astrofisico di Catania, Via S.Sofia, 78 - 95123 Catania, Italy

[3] Casaccia Research Center (CAMU), P.O. Box 2004, Roma, Italy

Abstract. A practical utilization of the Astrocomp web portal is illustrated in the framework of the numerical simulations of Astrophysical systems.
It is shown how to handle the various aspects related to the performing and managing of a typical N-body simulation. For this purpose, the features and usage of one of the parallel codes available in AstroComp, namely the "treeATD" code, are briefly described.

Key words. Numerical methods – Computer simulations – Galactic dynamics

1. Introduction

AstroComp is a project that aims at creating a portal for allowing the use and handling of high-performance numerical tools for Astrophysics, on a grid of supercomputers. The goal is also to set up a repository of computational codes and common databases in order to make them available and enjoyable, with a user-friendly graphical interface, to the entire national (and international) community. It is thus possible to benefit by the use of many different numerical tools implemented on high performance computing resources (HPC), both for theoretical astrophysics and cosmology and for the storage and analysis of astronomical data, without the need of specific training, know-how and experience either in computational techniques or in database construction and management methods. In the following, we describe the basical feature of the portal and, in some detail, one of the N-body code presently available to any registered user.

2. Present project status and description

A prototype of the portal (see Fig. 1) is already working and can be visited

Send offprint requests to: P. Miocchi
Correspondence to: P.le A. Moro, 2 - 00185 Roma

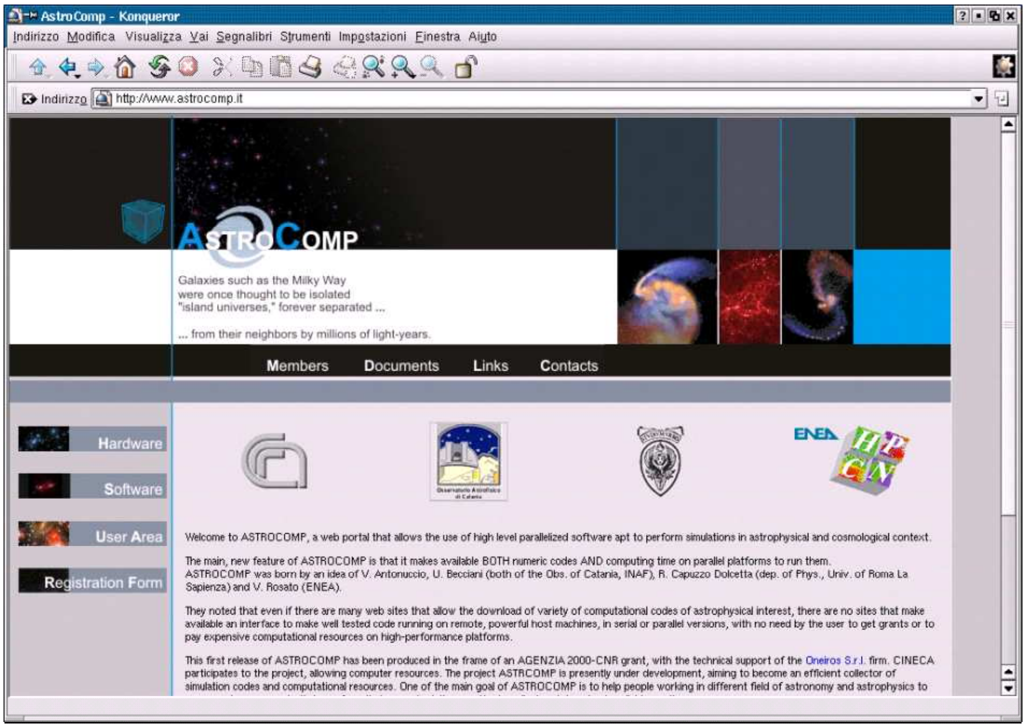

Fig. 1. *Home page of the present version of the AstroComp web portal.*

at `http://www.astrocomp.it/`, where further details on the project can also be found. At the present stage, AstroComp hosts three N-body codes plus a visualization tool (see Fig. 2). The codes are: i) the treeATD code, developed by P. Miocchi and R. Capuzzo Dolcetta, see Miocchi & Capuzzo Dolcetta (2002); ii) the FLY code, developed by V. Antonuccio and U. Becciani, see Becciani & Antonuccio-Delogu (2001); and iii) HSNbody, a direct-summation code which is presently under implementation into the portal, see Capuzzo Dolcetta et al (2001). TreeATD and FLY have been already tested and fully working on the server.

Anyone who is interested in using the provided facilities has to do a user registration first. The steps to do that are as follows:

1. register as a new user by clicking on the `Registration From` button on the left-hand side of the home-page;
2. fill in carefully the form and submit it;
3. wait for a confirmation e-mail sent by the portal administrator.

Clicking on the `User Area` button, any registered user can have the access to the computational facilities of AstroComp, after the login procedure.

At present, an AstroComp user is able to:

- start a new simulation by choosing which code has to be used and by determining the simulation parameters via the `Parameters` on-line form (with initial conditions that can be uploaded to the server);
- choose among different platforms from the pool of the available resources of AstroComp, taking into account the

work-load and the accessibility of each system;

or

- browse the status of a previously launched job, possibly checking the intermediate results with a preliminary 'on-the-fly' visualization tool already available in the portal itself (Fig. 2);
- download the final and/or intermediate results.

Moreover, anyone who visits the portal (not necessarily a registered user) can also examine all the implemented software (clicking on the `Software` button on the home-page), read the enclosed documentations and manuals, and take a look at the features and the status of all the computing machines that can be employed (clicking on `Hardware`).

3. TreeATD Description

'TreeATD' (tree-code with Adaptive Tree Decomposition) is a parallel tree-code for the computation of the gravitational interactions among the particles of a self-gravitating system. The algorithm used in this code, an improvement of the method firstly described in Barnes & Hut (1986) (see also Miocchi (1998) and Miocchi & Capuzzo Dolcetta (2002)) makes such evaluation faster, reducing the computational time from $O(N^2)$, typical of "direct-summation" methods, to $O(N \log N)$.

3.1. Main features

3.1.1. Tree-construction

At the beginning, a cubic box enclosing the whole system (the *root* cell) is found. This box is then subdivided into 8 cubic sub-boxes and the subdivision goes on recursively for each sub-box up to boxes with only one particle inside (terminal boxes). In the next stage, the relevant multipolar coefficients (total mass, centre-of-mass position and quadrupole moment) are computed for all boxes. These coefficients will be used to evaluate the force acting on a particle which is sufficiently far from a given box. The logical pointed structure which is built this way, corresponds to an octal-tree graph, which the name of the algorithm comes from.

3.1.2. Tree-walking

To compute the force acting on a particle, the algorithm "climbs" the tree starting from the root box, in the following recursive way: if the particle is sufficiently distant from the box, then the gravitational force due to all the particles it contains is computed through a multipolar expansion (truncated at the quadrupole) determined by the coefficients evaluated during the previous phase. This is done only if the cell, having size l and center-of-mass at a distance d from the particle, obeys to the 'open angle' criterion, i.e. if $l/d < \theta$, with $\theta > 0$ the 'tolerance parameter' set by the user. Otherwise, the algorithm goes to the next upper level of the tree and repeats recoursively this procedure for the sub-boxes. If a terminal box is met, then the force is evaluated by a direct and trivial summation.

3.1.3. Parallelization

Depending on the architecture on which the version of treeATD has to run, we followed slightly different parallelization approaches, in particular depending on whether the platform has a shared or distributed memory structure. In any case, the code adopts an original parallelization algorithm for the tree-construction phase, that can be outlined as follows:

- Firstly, after a 'random' distribution of the particles to the processors, the largest boxes are constructed and their multipolar coefficients evaluated, starting from the 'root' box, as peviously mentioned. The processors work simultaneously on the same boxes, dealing with different sub-sets of particles.

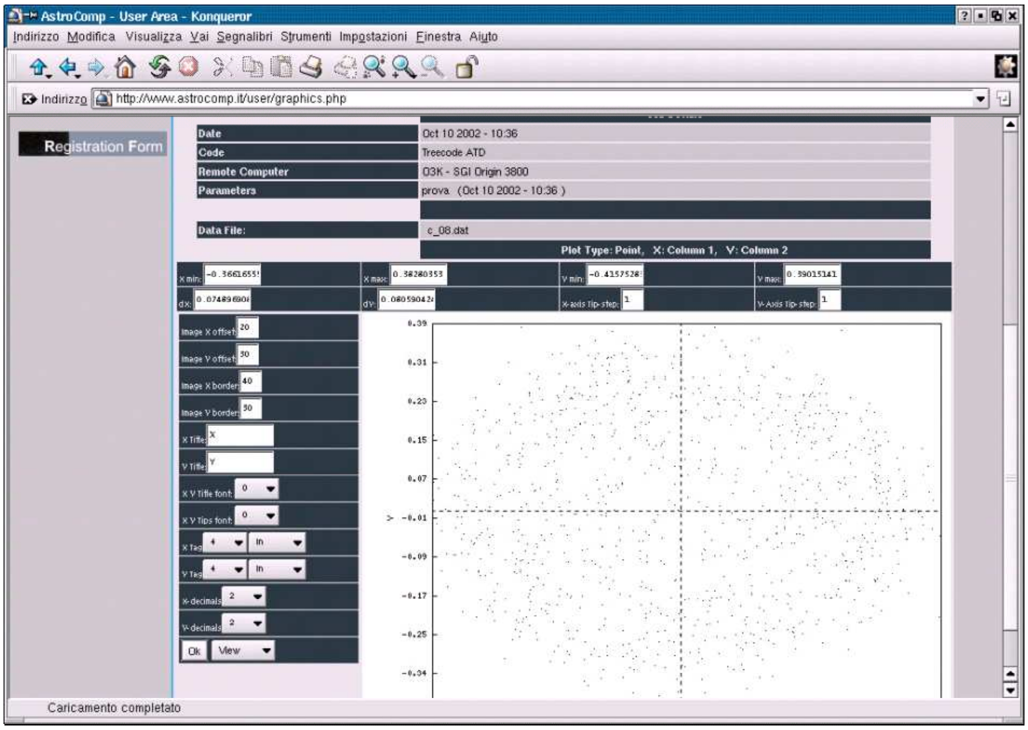

Fig. 2. *Example of utilization of an on-line visualization tool in AstroComp.*

Given n particles in the box, the load-balancing is guaranteed by $n/p >> 1$. For this reason, this stage involves only boxes with $n > kp$ where $k \sim 10$ is a parameter given a priori.

- Then, all the largest boxes with $n \leq kp$ are distributed to the processors which perform the tree-construction working independently of each other in an asyncronous way. That allows to reach a good load-balancing, thanks to the large number of boxes in the 'upper levels' of the tree structure, in comparison with the number p of processors.

In a distributed memory context this parallel tree construction allows also to build an efficient domain decomposition which minimizes inter-processors communications, see Miocchi & Capuzzo Dolcetta (2002) for further details.

3.2. Performances

In the Table 1 the CPU-time is shown in the case of the implementation on the Origin 3800, for one computation of all the forces acting among 10^5 bodies of a system distributed according to a non-uniform spherical density distribution (Plummer model). For a fixed number of processors (1,2,4 and 8) the CPU-time (in seconds), the computational velocity (in particles per second) and the speedup ratio are written (the speedup indicates the ratio between the computational time spent by a single processor and that spent by p processors).

We can see that the code shows a good performances scaling for up to 4 processors, while it degrades for $p = 8$ because this implementation is suitable for a shared memory context (in the Origin 3800 the memory of 1 single node is shared among 4 processors only).

Proc. no.	CPU-time (sec)	velocity (part/sec)	Speedup
1	53	1886.79	1
2	27	3703.7	1.96
4	14	7142.86	3.79
8	31	3225.81	1.71

Table 1. Velocity and speedup of treeATD on an Origin 3800 machine.

3.3. How to use treeATD

The code needs the input files shown in Table 2, plus some input parameters given as follows (in bracket: values related to a typical simulation, as an example):

- Number of particles, N (100,000)
- Time-scale, t_d (1.)
- Tolerance parameter, θ (0.7)
- Number of total steps (200)
- Steps in a time-scale (50)
- Binning factor, β (50)
- Smoothing radius, ϵ (0.002)
- Number of steps between two outputs, N_s (10)
- Maximum bin, b_{max} (14)
- Gravitational constant, G (1.)

The time-scale, is just a reference time the code uses as unit for determining the duration of the simulation and the maximum time step used for the time integration of particles trajectories. It should be set equal to the time-scale of the dynamical processes being studied. For example, for an isolated stellar cluster, the user may choose to fix $t_d \sim (G\rho)^{-1/2}$, where ρ is the half-mass density of the cluster. Of course the numerical value of such time depends on the used units, which in turn depend on the value given to G and on the units used for the positions and velocities in `coords.dat`.

The number of total steps determines how long is the simulation in terms of the time step Δt, this latter being the maximum step that the code uses in the integration of the equations of motion and that the user determines through the fraction $\Delta t \equiv t_d/$'steps in a time-scale'. The code adopts an improved leap-frog algorithm with the block-time scheme (see Aarseth (1985), Eastwood & Hockney (1988)), so that for each particle of the system an individual and variable time-step is used. The time-step of the i-th particle is determined first by evaluating a 'local' time-scale depending on its kinematic properties computed in the first-neighbour reference frame, as $\tau = \min\{(d/a)^{1/2}, d/|\mathbf{v}|\}/\beta$ where d is the distance from the first-neighbour, $\mathbf{v}$ is the relative velocity, a is the modulus of the acceleration and β is the 'binning factor'. Then, the actual time-step is fixed as the closest time to τ having the form $\Delta t_i = \Delta t/2^b$, with $b = 0, 1, 2, ..., b_{max}$ the so-called *bin* of the particle. The original block-time scheme has been improved in order to preserve the third order in the local integration error, also during the change of a particle time-step, when a lost of time simmetry occurs.

The smoothing radius ϵ is the distance below which the gravitational potential is no longer keplerian but 'smoothed' according to a β-spline function (see Miocchi (1998), Hernquist (1987)), in order to avoid the singularity as $r \to 0$.

Every N_s steps Δt, the code produces and possibly saves data about the state of the simulation, according to Table 3. The file `snap.dat` contains, in each line, two integers, `s` (that must be a multiple of N_s, and in increasing order) and `i`, plus a string `A` 2 characters long. At each time $t =$`s`Δt (where `s` is read everytime from `snap.dat`) during the simulation, the program 'take a picture' of the system state writing coordinates, velocities and masses in the file `c_A.dat`, using the same format of `coords.dat`. Moreover the file `b_A.dat` is created (its j-th line contains the bin of the j-th particle). Only the data related to the j-th particle with $j = 1$,`i`+1, 2`i`+1, ... are saved; thus only if `i`$= 1$ all the particles data are stored. It is possible to stop the list of the saved system states by writing `s`< 0. Moreover, every N_s steps various informations are written: i) (on the standard out-

Input files	Format	Description
`coords.dat`	**7(1X,G15.8)**	Initial conditions of the system. Each line contains: the 3 spatial coordinates, the 3 velocity components, the mass of each particle.
`snap.dat`	**1X,I3,A2,I3**	Each line contains the time step at which the user wants a “picture” of the system, a string of two letters and an integer i.

Table 2. Input files needed by treeATD. The file format is expressed in Fortran notation.

Output files	Format	Description
`c_A.dat`	**7(1X,G15.8)**	Each line contains: the 3 coordinates, the 3 components of velocity, the mass of each particle, at the time labelled by the string `A`.
`b_A.dat`	**(1X,I6,1X,I2)**	Each line contains the number that identifies the particle and its bin, at the time labelled by the string `A`.
`salva.dat`	binary	status of the simulation

Table 3. Output files produced by the code. The format is in Fortran notation.

put) the time achieved by the simulation (in t_d unit), the maximum level reached in the tree, the side of the cell that contains all the particles and the maximum bin of the particles, ii) (on the file `salva.dat`) all the relevant variables and parameters that characterised the state of the simulation at that time.

At the beginning of a run, the user can choose whether continuing a previously stopped (or crashed) simulation. In this case, the program reads from the file `salva.dat`, the last stored state of the previous interrupted simulation; then it asks for how many steps Δt it has to be continued. In the other case, initial conditions and parameters are normally read from the files above-described

The periodic saving of the simulation status file, permits, on one side, to re-start a crashed or interrupted simulation, and, on the other side, to subdivide a long simulation in shorter “sub-simulations” which can be executed on system queues with restrictive computational time limits.

Finally, if the user wants to re-start a simulation that gave not satisfactory results (for example with a too large error on the total energy conservation), the latest created file (among the various `c_A.dat`) that satisfies a given ‘qualitative’ criterion can be used as initial conditions (obviously this can be done only if the saving of the whole system has been selected): it will be sufficient to copy `c_A.dat` in `coords.dat`. On the other hand, the file `snap.dat` will have to be modified, because the time labelled with `A` is in this case the new initial time and the user may wish to change the value of, e.g., Δt in order to improve the results of the simulation.

4. Future perspectives

One of the next aim of AstroComp project is to extend the “computational environment” managed be the web portal to a wider and wider pool of high-performances platforms, making them actually available to all the researchers that use, or would like to use, HPC resources, giving the chance to employ the most common and valid software for astrophysical and cosmological simulations, as well as for data analysis.

Many improvements are planned. They regard:

- the implementation of on-line tools for the automatic generation of the most used and relevant initial conditions for astrophysical/cosmological simulations.

- the implementation of scheduling methods for the management of very long-term simulations by subdividing them in shorter "sub-simulations" that fit in the strict CPU-time limits of many job queue systems.
- the inclusion of advanced data analysis and storage tools (e.g. AstroMD, see http://www.cineca.it/astromd/).

Acknowledgements. Part of this work was supported by CNR *Agenzia 2000* and by the INAF-CINECA consortium under grant *inarm014* and *cnarm12a*.

References

Aarseth, S.J., 1985, in Multiple time scales, (Acad. Press), 378

Barnes, J.E., Hut, P., 1986, Nature, 324, 446

Becciani, U., Antonuccio-Delogu V., 2001, Comp. Phys. Comm., in press (astro-ph/0101148).

Capuzzo Dolcetta, R.,Pucello N., Rosato V., Saraceni F., 2001, Journ. of Comp. Phys., 174, 208.

Eastwood, J.W., Hockney, R.W., 1988, Computer Simulation using Particles, (Bristol: Hilger)

Hernquist, L., 1987, ApJS, 64, 715

Miocchi, P. 1998, PhD thesis in Physics, Universitá di Roma "La Sapienza".

Miocchi, P., Capuzzo Dolcetta, R., 2002, A&A. 382, 758.