Bull. Nov. Comp. Center, Num. Anal., 17 (2015), 69–78

A domain decomposition algorithm using SPH and PIC methods for simulating gas-dust gravitating disks*

N.V. Snytnikov, O.P. Stoyanovskaya

Abstract. We present a new parallel algorithm for supercomputer simulation of gas-dust circumstellar gravitating disks. The algorithm uses the domain decomposition technique and combines numerical methods of smooth particle hydrodynamics (SPH), particle-in-cell (PIC) and grid-based gravitational solver with the convolution method and parallel multidimensional Fast Fourier Transform.

The algorithm is designed to address the specificity of simulating non-stationary rotating disks, where the matter, represented by SPH and PIC particles, moves across a computational domain, and therefore requires efficient load-balancing methods. The parallel algorithm is suitable for running on supercomputers with different types of architectures: standard (CPU-based), and hybrid (CPU with NVIDIA GPU and Intel Xeon Phi).

1. Introduction

Supercomputer simulation of the dynamics of gravitating systems (such as galaxies and circumstellar disks) is one of the biggest challenges in the modern computational astrophysics. On the one hand, numerical methods aimed at simulating these processes were developed rapidly over the past 2–3 decades. A variety of sophisticated algorithms were proposed. Among them are: smooth particle hydrodynamics [1–4], particle-in-cells [5,6], treecode [7], piecewise parabolic method [8], adaptive mesh refinement [9] and others. On the other hand, existing parallel implementations of these methods are not suitable for an extensive set of astrophysics problems and usually focus on one particular astrophysics problem like well-known cosmological simulations (Millenium [10], Bolshoi [11], BlueTides [3] and others). Parallel implementations rely on some special conditions that are typical of a particular problem, but could hardly demonstrate good performance in other contexts.

Another important challenge is the requirement to develop algorithms that could be used for different types of supercomputer architectures: con-

*Supported by the RFBR under Grants 14-01-31088, 14-01-31516, and 14-07-00241. The ministry of Education and Science of the RF. Numerical experiments were conducted at the Siberian Supercomputer Center, Joint Supercomputer Center, and MSU Lomonosov Supercomputer.

ventional (comprising CPUs only) and hybrid (comprising of CPUs and massively parallel processors like NVIDIA GPU and Intel Xeon Phi).

In this paper we propose a parallel domain decomposition algorithm for the earlier developed numerical model of gravitating circumstellar disks [12, 13]. Currently we are focusing mainly on the model of a thin disk (where the movement of the matter in the vertical direction is neglected), although the proposed parallel algorithm can be used in 3D simulations without any restrictions.

The novelty of our approach is a method of dynamic load-balancing for SPH and PIC particles (which move among computational subdomains). This method uses features of numerical methods and underlying astrophysical problem.

The model of a thin circumstellar disk includes the gasdynamics equations for surface density, which are solved with the SPH method:

$$\sigma_{\text{gas}} = \int_{-\infty}^{+\infty} \rho_{\text{gas}}\, dz; \quad p^* = \int_{-\infty}^{+\infty} p\, dz.$$

$$\frac{\partial \sigma}{\partial t} + \operatorname{div}(\sigma \boldsymbol{v}) = 0, \quad \sigma \frac{\partial \boldsymbol{v}}{\partial t} + \sigma(\boldsymbol{v}, \nabla)\boldsymbol{v} = -\nabla p^* - \sigma \nabla \Phi,$$

$$\frac{\partial S^*}{\partial t} + (\boldsymbol{v}, \nabla) S^* = 0, \quad p^* = T^* \sigma.$$

Here we use the notations: $\boldsymbol{v}$ is the velocity of gas in a disk plane, p^* is the surface pressure of gas, γ^* is the effective polytropic exponent for quasi-3D model that has the following relation with γ: $\gamma^* = 3 - \frac{2}{\gamma}$, $T^* = \frac{p^*}{\sigma}$ and $S^* = \ln \frac{T^*}{\sigma^{\gamma^*-1}}$ are variables corresponding to the gas temperature and entropy, and $\boldsymbol{a} = -\nabla\Phi$ is the acceleration of particles in the external and self-consistent gravitational field.

The gravitational potential Φ is defined as a sum of the central body potential and the disk potential:

$$\Phi = \Phi_1 + \Phi_2, \quad \Phi_1 = -\frac{M_c}{r},$$

where M_c is the mass of the central body, the potential Φ_2 of the self-consistent gravitational field satisfies the Poisson equation:

$$\Delta \Phi_2 = 4\pi \sigma_{\text{gas}}, \quad \Phi_2 \xrightarrow[r\to\infty]{} 0.$$

The dust phase dynamics of a circumstellar disk is defined by the Vlasov equation. It is solved with well-known particle-in-cell method (PIC) [6]:

$$\frac{\partial f}{\partial t} + \boldsymbol{u}\frac{\partial f}{\partial \boldsymbol{r}} - \nabla\Phi\frac{\partial f}{\partial \boldsymbol{u}} = 0, \quad f(0, \boldsymbol{r}, \boldsymbol{u}) = f^0(\boldsymbol{r}, \boldsymbol{u}).$$

Here $f(t, \boldsymbol{r}, \boldsymbol{u})$ is a distribution function of the matter (dust in circumstellar disks), $\Phi(t, \boldsymbol{r})$ is the gravitational potential, and $\rho(t, \boldsymbol{r})$ is the density, to be calculated using the following equation:

$$\rho(t, \boldsymbol{r}) = \int_{\boldsymbol{u}} f(t, \boldsymbol{r}, \boldsymbol{u})\, d\boldsymbol{u}.$$

The initial densities of gas and dust are defined as a Mclauren disk with the mass M_{gas} and radius R:

$$\sigma_{\text{gas}}(r) = \frac{3M_{\text{gas}}}{2\pi R^2}\sqrt{1 - \frac{r^2}{R^2}}.$$

The gas temperature at the initial moment is defined as $T^*(r) \sim \sigma(r)$ the temperature T_0 at the center of the disk.

The numerical method for solving this system of equations can be briefly described in the following way:

1. Introduce a uniform grid in the Cartesian computing domain, and fill in it with SPH particles and PIC particles according to the gas and dust densities.
2. Solve the gas-dynamics equations using SPH method: compute the new coordinates of SPH-particles and the corresponding gasdynamics parameters. This step requires that each particle have links to its neighbor particles (which are not necessarily located in the same grid cell), therefore some kind of particles sorting and searching procedure is needed.
3. Solve the Vlasov equation using the PIC method. Since PIC-particles are interacting with each other through the grid-based gravitational field, their trajectories can be interacted independently in an arbitrary order. However, it is much more efficient to process particles in a cell-by-cell order, because of using a fast cache memory of modern CPU/GPU/Phi processors (all values of a cell will be loaded in the cache memory only once). Thus, special data structures for the PIC particles are also needed.
4. Compute the grid-based gravitational potential using the convolution method with Fast Fourier Transform.

A parallel algorithm should address the two issues: (a) to divide the computational domain into subdomains to make sure that grid functions for each subdomain can be calculated by a single processor with 1–4 GB of memory, and (b) to distribute particles among processors in such a way

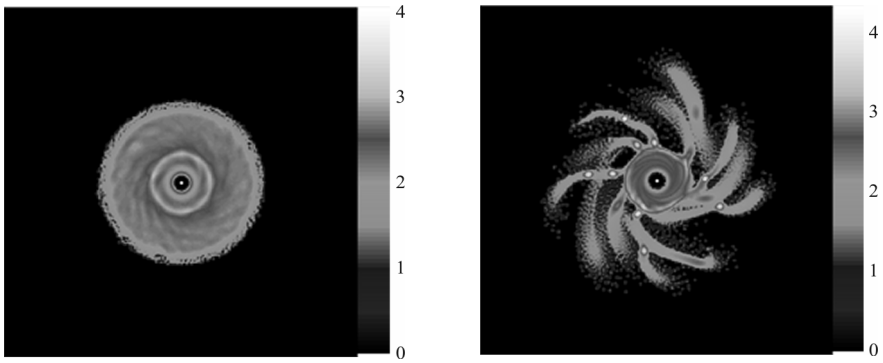
that particles could access values of the grid function, freely move among subdomains, and the total amount of particles transferring at each step among subdomains be minimal.

In Section 2, we give a description of the proposed parallel domain decomposition method. The parallel algorithm of the gravitational potential calculation is discussed in Section 3.

2. Parallel domain decomposition

The main difficulty in developing a parallel algorithm is how to couple a numerical method of integrating individual trajectories of modeling particles with grid-based methods for calculating density, gravitational potential and forces. Typically, a maximal number of grid nodes to be placed into a single computational processor (CPU, GPU, Phi) is equal to 8–16 million ($256 \times 256 \times 256$ for a 3D problem and 4096×4096 for a 2D problem). This means that larger computational domains ($1024 \times 1024 \times 1024$, for example) should be divided into smaller subdomains. At the same time, SPH and PIC particles may easily cross the initial domain many times during a single computation (for example, if we consider the simulation of disks on the timescale of dozens rotations, where each particle has an elliptic or epicyclic orbit around the center of mass).

Another problem arises because particles are non-uniformly distributed among subdomains. A rotating disk may develop small and large clumps, spiral waves, and other transient or secular structures that move over a computational domain in an unpredictable manner. This implies that the number of particles in each subdomain may differ up to orders of magnitude. In addition, their number may drastically change over time: for example, when one of the clumps containing a large number of particles starts the rotation around the central body.



Simulation of gravitational instabilities in a circumstellar disk using SPH method and grid-based gravitational calculation

A general scheme of the new algorithm, that addresses all the mentioned issues, is described below. It uses the following assumptions:

- computational complexity of solving the Vlasov equation and the hydrodynamics equation (i.e. integrating SPH and PIC particle trajectories) is greater (up to 10 or 100 times) than the computational complexity of calculating the gravitational potential. This assumption is justified because the complexity of calculating the gravitational potential scales as $O(n \log n)$ (where n is the total number of nodes), while a typical number of PIC particles per one grid cell (a grid node) is usually within the range of $10 \div 100$. Also, the number of arithmetic operations per each particle is greater than the number of operations per one cell (a grid node). For SPH particles the situation is similar, although the number of particles per grid cell can be taken within the range of $1 \div 10$.
- each particle should satisfy the Courant condition (otherwise, the simulation becomes numerically unstable): this means that a particle may travel less than half of a minimal cell dimension during one time step. In fact, for overwhelming the majority of particles the speed is much lower. This implies that the number of particles, that should be transferred between the adjacent subdomains is much less than the total number of particles in each subdomain. This number can be estimated as $10n_y$ for a 2D problem and $10n_yn_z$ for a 3D problem, where coefficient 10 is the estimated number of migrating particles and n_y, n_z are the number of boundary grid nodes.

Based on these assumptions we have developed the algorithm of dynamic load-balancing, which transfers particles between processors and corresponding computational domains and provides a moderate amount of communications. This algorithm is in the process of implementation, and preliminary results and experiments have shown good scalability at least up to a thousand of processors. For simplicity, we split the algorithm to several procedures: Algorithm 1 is a general scheme with references to the subroutines described as Algorithm 2–6.

Algorithm 1: *A general scheme of parallel domain decomposition.* The initial computational domain (2D or 3D) is uniformly subdivided into K subdomains $S_1, S_2, \ldots, S_K$ in X direction. A group of processors G_k is assigned to each subdomain S_k. The number of processors in the group G_k is equal to P_k, $P_k \geq 1$. The total number of processors is equal to $P = \sum P_k$. The number of nodes (cells) in each computational domain is chosen in such a way, that all subdomain grid functions used in the PIC and SPH methods (gravitational potential, density, force, pressure, etc.) should be entirely located in memory of one processor with a sufficient room to store

arrays of particles (i.e their spatial coordinates and velocities). Usually the number of nodes is equal to $0.5 \div 2$ million nodes (128^3 or 1024^2). The number of processors in the group G_k is chosen to provide approximately a uniform number of particles per processor (see Algorithm 2). This means that if the subdomain S_{k_1} contains more particles than the subdomain S_{k_2}, then the number of processors P_{k_1} in the group G_{k_1} will be greater or equal to the number P_{k_2} in the group G_{k_2}. Each group of processors G_k and the corresponding subdomain S_k has its main processor g_k^0, that cannot be reassigned to any other subdomain in the coarse of simulation (even if the subdomain S_k does not contain any particles).

1. At the beginning of the simulation (the time instant $t = 0$):
   (a) for each subdomain S_k compute the total number of particles, the corresponding number of processors in a subdomain group, and the number of particles per each processor (see Algorithm 2).
   (b) allocate arrays in each processor and generate the spatial coordinates and velocities.
   (c) populate PIC and SPH particles in the computational domain according to the user-defined density functions and distribute the particles among processors.
2. Compute the density grid functions of PIC- and SPH-particles (see Algorithm 3).
3. Compute the gravitational potential using the parallel convolution method (see Algorithm 6). At this step, only the main processors g_k^0 from each group G_k are used. The rationale outside this approach is that the computation of gravitational potential takes much less time than computation of particles, so it can be accomplished with a fewer number of computational resources in a reasonable time. If hydrid computers (CPU and GPU) are used, then this step is done only on CPUs, while GPUs are reserved for integrating trajectories of particles. (Anyway, it is possible to make a massive parallelization of this step, using all the processors.)
4. Redistribute the computed potential grid function from the main processor g_k^0 to all other processors g_k^i of the group G_k.
5. For each processor compute the new coordinates of SPH and PIC particles for the next time step (see Algorithm 4).
6. Compute a new number of particles that should be located in each subdomain S_k (this information is transferred among processors). Compute the number of processors that should be assigned to each subdomain (see Algorithm 2). Redistribute particles among the new processor groups (see Alorithm 5).

7. Increment the time step and go to Step 2.

Algorithm 2: *Calculating the number of processors in a group.* Let us assume that each subdomain S_k contains N_k particles (these values are known beforehand), and the total number of particles $N = \sum N_k$. We have to figure out how many processors should be assigned for treating N_k particles to provide a uniform load of processors. Each processor group contains at least one (the major) processor g_k^0 even if the number of particles in this subdomain is small enough. At each time step we have to calculate number $N_{\max}$: this is a maximum number of particles per processor. If a processor contains exactly $N_{\max}$ particles, then the computer costs for this time step will be defined by this processor. So, our task is to make $N_{\max}$ as minimal as possible. We use a bisection algorithm to find this value:

1. Compute values $N_{\text{low}} = N/P$ and $N_{\text{upp}} = 2N/P$. One can see that N_{low} is less than the desired $N_{\max}$ and N_{upp} is greater than the optimal $N_{\max}$. Assign $N_{\max} := (N_{\text{low}} + N_{\text{upp}})/2$.
2. Calculate the values $P_k = (N_k/N_{\max}) + 1$. Calculate the value $P^* = \sum P_k$. Compare P^* with P:
   - If $P^* < P$, then $N_{\text{upp}} := N_{\max}$, go to Step 3.
   - If $P^* > P$, then $N_{\text{low}} := N_{\max}$, go to Step 3.
   - If $P^* = P$, then $N_{\text{upp}} := N_{\max}$, check whether $N_{\text{upp}} - N_{\text{low}} < N_{\text{eps}}$ (where $N_{\text{eps}} \ll N_{\max}$ and is defined by the user), then go to Exit, otherwise go to Step 3.
3. Assign $N_{\max} := (N_{\text{low}} + N_{\text{upp}})/2$ and go to Step 2.

The density and integrating trajectories of particles are computed using a usual particle-in-cell algorithm. However, for parallelization it is important to have appropriate data structures combining cells and particles. Algorithms 3-4 below outline properties of these data structures.

Algorithm 3: *Computation of the density inside subdomains.*

- Particles are sorted in each subdomain, where the sorting key is a serial number of a cell (firs, by ordering by the coordinate X, then by Y and, finally, by Z). This kind of a sorting algorithm has a linear complexity and can be done when flying while integrating the particle trajectories (see Algorithm 4).
- Then particles are redistributed among processors of this group according to their serial number in a sorted array.
- The sorting procedure provides an efficient use of the processor cache memory, while calculating the coordinates of PIC-particles and a fast search for the neighbors in SPH method.

- It is required to keep ghost boundary cells for SPH particles on each processor (these cells are boundary cells from the adjacent subdomain and require computing hydrodynamic parameters).
- The density grid function is calculated using PIC kernel (multilinear interpolation to the cell nodes). And after that, it is gathered as a grid function on the processor g_k^0.

Algorithm 4: *Computation of particles coordinates* (*trajectories integration*).

- A special data structure is developed: a cell with a linked array of particles that are currently located in this cell.
- For each cell, integrate the trajectories of particles that are linked with a current cell.
- Particles that move to other cells are to be linked with their new cell, while all links with a previous cell are erased.
- Particles that move to the cells located outside of a current subdomain (or outside of the cells assigned for this processor) are marked as external particles and will be moved to another subdomain (or a processor) (see Algorithm 5).

Algorithm 5: *Particles redistribution among subdomains after the new coordinates computation.*

1. After applying Algorithm 4, each processor has the three types of particles: particles that are kept inside a current subdomain, particles that are to be transferred to the left adjacent subdomain, and particles that are to be transferred to the right adjacent subdomain. The number of transferred particles is not really big because of the Courant condition: particles cannot cross the boundary of more than one cell at one time step.
2. Compute a new number of particles for each subdomain (this can be done with Algorithm 4 and transfer this information among processors).
3. Compute the number of processors for each subdomain (see Algorithm 2).
4. Determine processors of each group that should be transferred to other subdomains (to other groups). Transfer particles that should be kept on a current subdomain.
5. Transfer other particles from the processors of this subdomain to the processors of the adjacent subdomain.

3. A parallel algorithm for calculating the gravitational potential

We use a parallel version of the convolution method proposed by Hockney. The sequential version is described in details in [5,12], while the present paper provides a brief description of the parallel algorithm. The 2D algorithm is essentially the same as in the 3D case with the only difference of omitting the steps with applying Fast Fourier Transform to the direction Z.

Algorithm 6: *A parallel convolution method in the 3D case.*

1. A 2D or a 3D computational domain is divided into subdomains in the direction X.
2. Apply Direct Fast Fourier Transform to the density grid function and the kernel grid function in direction Y.
3. Apply Direct Fast Fourier Transform in the direction Z.
4. Make a transposition of slabs from Y- to X-direction.
5. Apply Direct Fast Fourier Transform in the direction X.
6. Multiply the transformed kernel grid function by the transformed density grid function. Store the multiplication result (MR).
7. Apply Inverse Fast Fourier Transform in the direction X to the MR.
8. Make an inverse transposition of slabs of MR from X- to Y-direction.
9. Apply Inverse Fast Fourier Transform in the direction Z.
10. Apply Inverse Fast Fourier Transform in the direction Y.

As one can see, the only interprocessor communications are the direct and inverse transpositions (that correspond to MPI Alltoall communication). The numerical experiments have shown that for moderate grids and a corresponding number of processors (about 1024^3 nodes and $256 \div 1024$ processors) the time spent on communications is acceptable (about 30 % of the total time spent on the gravitational potential calculation), so the calculation takes less than 1 second.

The algorithm was implemented using the FFTW library [14], and the results of the computational experiments (performed in Siberian Supercomputer Center, Joint Supercomputer Center, and the Lomonosov supercomputer in MSU) is to appear [15].

4. Conclusion

We have developed parallel domain decomposition algorithm for simulating two-phase (gas-dust) circumstellar gravitating disks with the SPH and the PIC methods. The software implementation will be used to study processes in gravitationally unstable protoplanetary disks and to develop of high-density clumps with a fine resolution and acceptable time steps.

References

[1] Gingold R.A., Monaghan J.J. Smoothed particle hydrodynamics — theory and application to non-spherical stars // Monthly Notices of the Royal Astronomical Society. — 1977. — Vol. 181. — P. 375–389.

[2] Springel V., Yoshida N., White S.D.M. GADGET: a code for collisionless and gasdynamical cosmological simulation // New Astronomy. — 2001. — Vol. 6. — P. 79–117.

[3] Feng Y., Di Matteo T., Croft R.A.C., et al. BlueTides: First galaxies and reionization // Monthly Notice of Royal Astronomical Society. — 2015 (to appear).

[4] Pearce F.R., Couchman H.M.P. Hydra: a parallel adaptive grid code // New Astronomy. — 1997. — Vol. 2, Iss. 5. — P. 411–427.

[5] Hockney R.W., Eastwood J.W. Computer Simulation Using Particles. — New York: McGraw-Hill, 1981.

[6] Berezin Yu.A., Vshivkov V.A. Particle-in-Cell Method in the Plasma Dynamics. — Nauka, 1980.

[7] Barnes J.E., Hut P. A hierarchical $O(N \log N)$ force-calculation algorithm // Nature. — 1986. — Vol. 324. — P. 446–449.

[8] Colella P., Woodward P.R. The Piecewise Parabolic Method (PPM) for gasdynamical simulations // J. Comput. Phys. — 1984. — Vol. 54, Iss. 1. — P. 174–201.

[9] Dubeya A., Antypasb K., Ganapathyc M.K., et al. Extensible component-based architecture for FLASH, a massively parallel, multiphysics simulation code // Parallel Computing. — 2009. — Vol. 35. — P. 512–522.

[10] Springel V., White S.D.M., Jenkins A., et al. Simulations of the formation, evolution and clustering of galaxies and quasars // Nature. — 2005. — Vol. 435. — P. 629.

[11] Klypin A.A., Trujillo-Gomez S., Primack J. Dark matter halos in the standard cosmological model: results from the Bolshoi simulation // Astrophys. J. — 2011. — Vol. 740. No.. 2. — P. 102.

[12] Stoyanovskaya O.P., Snytnikov N.V., Snytnikov V.N. An algorithm for solving transient problems of gravitational gas dynamics: a combination of the SPH method with a grid method of gravitational potential computation // Vychislitel'nye Metody i Programmirovanie. — 2015. — Vol. 16. — P. 52–60 (In Russian).

[13] Snytnikov N. Scalable parallel algorithm for solving the collisionless Boltzmann–Poisson system of equations // Astronomical Society of the Pacific Conference Series. — 2012. — Vol. 453. — P. 393.

[14] Frigo M., Johnson S.G. FFTW software. — http://www.fftw.org.

[15] Snytnikov N.V. Parallel algorithm for calculating gravitational potential of isolated systems // Computational Technologies. — 2016 (to appear).