Journal of Computational Physics **176,** 276–294 (2002)
doi:10.1006/jcph.2001.6978, available online at http://www.idealibrary.com on IDEAL®

Dust Dynamics in Protoplanetary Disks: Parallel Computing with PVM

Carlos de la Fuente Marcos,* Pierre Barge,† and Raúl de la Fuente Marcos*

*Universidad Complutense de Madrid, E-28040 Madrid, Spain; and †Laboratoire d'Astrophysique de Marseille, B.P. 8, 13376 Marseille Cédex 12, France
E-mail: nbplanet@ucmail.ucm.es, pierre.barge@astrsp-mrs.fr, rfuente@ucmail.ucm.es

Received July 13, 2001; revised November 5, 2001

We describe a parallel version of our high-order-accuracy particle-mesh code for the simulation of collisionless protoplanetary disks. We use this code to carry out a massively parallel, two-dimensional, time-dependent, numerical simulation, which includes dust particles, to study the potential role of large-scale, gaseous vortices in protoplanetary disks. This noncollisional problem is easy to parallelize on message-passing multicomputer architectures. We performed the simulations on a cache-coherent nonuniform memory access Origin 2000 machine, using both the parallel virtual machine (PVM) and message-passing interface (MPI) message-passing libraries. Our performance analysis suggests that, for our problem, PVM is about 25% faster than MPI. Using PVM and MPI made it possible to reduce CPU time and increase code performance. This allows for simulations with a large number of particles ($N \sim 10^5$–10^6) in reasonable CPU times. The performances of our implementation of the parallel code on an Origin 2000 supercomputer are presented and discussed. They exhibit very good speedup behavior and low load unbalancing. Our results confirm that giant gaseous vortices can play a dominant role in giant planet formation.

Key Words: accretion; accretion disks; MPI; parallel computing; planetary systems; PVM; solar system formation; turbulence.

1. INTRODUCTION

Astrophysics is a particularly fruitful field for the application of computer simulation because the systems under study—protoplanetary disks, minor bodies, planets, stars, and galaxies—are not adaptable to controlled laboratory experimentation. In astrophysics, experiments are mainly carried out by simulating a system on a computer and performing experiments with that simulation. Astrophysical particle systems divide into two main

0021-9991/02 $35.00

classes: collisional and collisionless. The dynamical evolution of collisional systems is driven by two-body (or higher order) interactions as in star clusters. In contrast, the motion of a particle in a collisionless system is predominantly controlled by the local mean gravitational field and scarcely influenced at all by the particular distribution of other particles in its neighborhood [1]. In fact, if one observes the motion of particles in a collisionless system, they may appear to drift through each other like ghosts as they respond to the mean field. The particles do not react with each other; there are neither sharp changes of direction nor the formation of binary or higher multiplicity systems.

The evolution of a collisionless system is determined by the mean mass density. In a purely gravitational collisionless system the distribution of the masses of individual particles has no influence on the evolution of the local mean quantities. Although the rotation curve of a protoplanetary disk is dominated by a central mass (the star), its evolution is strongly affected by drag forces. In this environment, the dynamical behavior of a particle depends on the ratio of its response time against drag to its orbital period but not on the collisions with other particles. For systems with a fixed total mass the binary collision rate is inversely proportional to the number of massive objects N into which mass is divided. If N is doubled, although the number of scattering centers is also doubled, the force between any interacting pair of particles is divided by four as force is proportional to the square of the mass. The net result is that the total collisional effect is halved. If the system is composed of a large number of massive objects, as in a protoplanetary disk where $N \approx 10^{26}$ metric particles, the interparticle force is only comparable to the solar attraction if the interparticle distance is about 1 mm, but the characteristic interparticle distance is several orders of magnitude larger. Besides, the drag force is much more important than the interparticle gravitational force. For these reasons young protoplanetary disks are described as collisionless systems. On the other hand, such disks are thought to be turbulent during a period of their lives so as to explain the mass and angular momentum evolution and the constraints deduced from meteoritic data. Barge and Sommeria [2] suggested that large-scale coherent structures grow in these turbulent disks by analogy with what happens in two-dimensional fluid dynamics in which organized structures are known to emerge from random turbulence in rotating shear flows (for example, Jupiter's Great Red Spot). They found that large-scale anticyclonic vortices can capture and concentrate very efficiently the solid material of a protoplanetary disk, and they claimed that these vortices could play a central role in the first stages of planet formation. Chavanis [3] improved on this research by using an exact vortex solution of the incompressible two-dimensional Euler equation and the epicyclic approximation. This early work has recently been continued by de la Fuente Marcos and Barge [4] using sequential simulations. In this paper we show that noncollisional problems like the one described above match up very well against parallel programming on multicomputer architectures in which all of the processors are essentially running replicated sets of the same instructions and just passing information as required by the computation.

This paper is organized as follows. In Section 2 we describe the hardware used in this research. The software making the parallelization feasible is considered in Section 3. The physical model included in the code is analyzed in Section 4. In Section 5 we describe the code as well as our parallelization approach. The results are presented in Section 6 and our conclusions are outlined in Section 7.

2. HARDWARE

Since the early 1980s, the two major parallel architectures—shared-memory multiprocessors and message-passing multicomputers—have proven optimum for different problems. The shared-memory machines have the reputation of being easier to program. The memory in this type of architecture is called shared memory because the data are stored as a single large structure, as on any conventional computer. However, as the number of processors increases, the probability of a conflict occurring between two or more processors simultaneously attempting to access the same data gets higher. In contrast, message-passing multicomputer architectures can be scaled up to tens of thousands of processors because the memory in this type of architecture is associated with each individual processor. As a result, the data set must be broken up, or decomposed, and distributed to the memory of each processor with a set of program instructions telling the processor what to do. Data are shared between processors in the form of packets, or messages, that are sent from one node's memory to another via circuitry.

During the past few years, the computer industry has been yearning for an extensible systems architecture to replace symmetric multiprocessor (SMP) systems. With an understanding of the limitations of bus-based architectures, Silicon Graphics Inc., (SGI) set out to develop a revolutionary computer architecture that would combine the best features yet overcome the performance obstacles and bottlenecks of SMP and massively parallel processor (MPP) designs. In 1996, SGI announced and started shipping the Origin line of servers, establishing cache coherent nonuniform memory access (CC-NUMA) as a viable server platform. This architecture is not reliant on a bus but rather is designed using a series of nonblocking crossbar switches as an interconnection fabric, essentially adding incremental I/O as the system configuration grows. The Origin 2000 is CC-NUMA programmable as shared memory (SM) with a mixture of MPPs and SMPs. This special architecture is called S^2MP (scalable SMP) and it uses distributed shared memory (DSM). The Origin 2000 machines are based on the architecture of the scalable node 0 system. Each node board contains two MIPS RISC 64-bit R10000 processors with associated cache and memory. In an Origin 2000 machine, a number of processing nodes are linked together by an interconnection fabric called NUMA-link (SGI CC-NUMA hypercube). The interconnection fabric is a mesh of multiple, dynamically allocable transactions, allowing connections to be made from processor to processor as needed. As processors are added to the system, the bandwidth between processors and local memory remains constant, allowing for truly scalable performance as the system grows.

3. PARALLEL PROGRAMMING WITH PVM

The parallel virtual machine (PVM) is a software package developed by Oak Ridge National Laboratory, the University of Tennessee, and Emory University. It was initially conceived to enable a heterogeneous collection of computers linked by a network to function as a single large parallel computer and to be used as a coherent and flexible concurrent computation resource. The individual computers may be shared with local memory multiprocessors, vector supercomputers, specialized graphics engines, or scalar workstations. The PVM project began in the summer of 1989 at Oak Ridge National Laboratory. Version 2 of PVM was written at the University of Tennessee and released in March 1991. During

the following year, PVM began to be used in many scientific applications. Version 3 [5] was a full rewrite completed in February 1993. During the past few years, SGI and Cray Research Inc. have taken that implementation and extended it in several ways. PVM is able to run on the SGI Origin family of supercomputers. This computer family runs the UNIX-like operating system IRIX. The message-passing toolkit (MPT) for IRIX is a software package that supports interprocess data exchange for applications that use concurrent, cooperating processes on a single host or on multiple hosts. Data exchange is done through message passing, which is the use of library calls to request data delivery from one process to another or between groups of processes. The MPT 1.5 package (used in this research) contains the following components and the appropriate accompanying documentation: (1) PVM, (2) message-passing interface (MPI) [6], and (3) logically shared, distributed memory (SHMEM) data-passing routines. The MPT version of PVM has enhancements to enable the use of POSIX shared memory, which provides greater flexibility and robustness than offered by the previously used IRIX shared arenas. Default communication is based on transmission control protocol (TCP) sockets between processes on the same system and between different systems. Transfer speeds are relatively slow when sockets are used as the mechanism for communication. The MPT version of PVM also provides alternative mechanisms for communication. The socket communication has been optimized to utilize high-speed network devices more effectively. PVM has been integrated with the network queuing environment (NQE) so that it is possible to use PVM within a batch job in isolation from other PVM jobs.

SGI provides versions of PVM to support a variety of Needs. These versions provide users with a single subroutine interface for message-passing programming; this interface is portable and a *de facto* standard. PVM is available from its developers as public domain software and is being made available as vendor-supported software by SGI and a number of other computer vendors. PVM is supported on all SGI systems. The PVM software system consists of a library and commands that support PVM [5]. The PVM software provided by SGI was developed specifically for each system on which it runs. It is feasible to use PVM to communicate among processes on a number of different computer systems. The following characteristics apply to all PVM system combinations: (1) the user building an executable file for use on an SGI system links with a single PVM library, (2) the same standard library syntax and behavior are supported, regardless of how PVM is used, and (3) the performance of PVM in different basic scenarios differs significantly; this difference influences the communications strategy that should be used.

The following PVM terminology is used in this work: *task*, the UNIX process that uses PVM for communications; *application*, a number of tasks running the same program; and *process*, the entity running on the IRIX operating system or another UNIX system. PVM comprises a daemon (**pvmd3**) and a library of PVM interface routines (**libpvm3.a**). The daemon starts up PVM and constructs a virtual machine. User programs written in C, C++, or Fortran access PVM through **libpvm3.a**.

The PVM software system has been used in many computational applications and for solving large-scale problems in science, industry, and business since its initial release in 1991. PVM has been used in astronomical data analysis software since 1996. Mo *et al*. [7] implemented an optical prescription retrieval code using PVM in a mixed architecture network to use at Space Telescope Science Institute (STScI) and Boden *et al*. [8] developed a massively parallel, spatially variant maximum likelihood image restoration algorithm, also at STScI, to process images from the Hubble space telescope (HST)

wide-field planetary cameras. PVM has recently been used in astrophysics to study cosmological problems. Their Cosmos code by Ricker *et al.* [9] has been implemented for parallel computers using the PVM library, and it features a modular design which simplifies the addition of new physics and the configuration of the code for different types of problems. Another recent application of PVM in an astrophysical context was developed by Viturro and Carpintero [10]. They developed a parallelized tree code to simulate collisions of galaxies by clusters of personal computers using PVM as message-passing library software.

4. PHYSICAL MODEL

Many newly formed stars appear to possess residual protostellar disks of solar system size, which are likely to originate from the collapse of slowly rotating cloud cores like those observed in nearby dark clouds. The later evolution of such disks governs both the final stages of star formation and the formation of planetary systems. For modeling the dust disk, we use a thin disk model of the nebula, in which the surface densities (both for gas and particles) and the temperature are given by decreasing power laws. Hydrostatic equilibrium in the vertical direction is also adopted. Under the standard assumption of hydrostatic equilibrium in the thickness H of the nebula, $H \approx C_s \Omega$, where C_s is the sound velocity and Ω is the Keplerian angular velocity. In such a protoplanetary disk, for a given particle there is competition between gas drag and inertia. Details about the nebula model used in the calculations are shown in the next section.

The motion equations for a solid particle submitted to the attraction of the star and to the friction drag of the nebula gas are

$$\frac{d^2\vec{r}}{dt^2} = -GM_0\frac{\vec{r}}{r^3} - \frac{\vec{u}}{T_s} + \vec{\Psi}, \tag{1}$$

where $\vec{u} = \vec{v} - \vec{V}_g$, $\vec{V}_g$ is the gas velocity, and $\vec{\Psi}$ is the contribution of the self-gravity of the disk. T_s is the stopping time, i.e., a characteristic friction time scale which depends not only on the mass and velocity of the particles but also on the distance to the star. The drag regime depends on the size of the particles relative to the mean free path of the gas molecules ($\lambda \approx m_H \mu / \rho_g \sigma_{H_2}$, $\sigma_{H_2} \approx 2 \times 10^{-9}$ m^2),

$$T_s = \begin{cases} \frac{\rho_p s}{\rho_g C_s} & \text{if } s \leq 9/4\lambda \text{ (Epstein regime)}, \\ \frac{8\rho_p s}{3\rho_g C_D u} & \text{otherwise (Stokes regime)}, \end{cases} \tag{2}$$

where ρ_g is the density of the gas, C_s is the thermal velocity, ρ_p is the density of the solid material, s is the radius of the particle, and C_D is a nondimensional coefficient which depends on the Reynolds number, $\mathrm{Re} = 2su\rho_g/\eta_g$ (η_g is the viscosity of the gas, $\eta_g = 1/2\rho_g \lambda C_s$). The thermal velocity is given by $\sqrt{8\kappa T/\pi\mu m_{\mathrm{H}}}$, where κ is the Boltzmann constant, μ is the mean molecular mass ($\mu \approx 2.34$), and m_{H} is the mass of a hydrogen atom. Following

Weidenschilling [11], the dimensionless drag coefficient is given by

$$C_D = \begin{cases} \frac{24}{\mathrm{Re}} & \text{if } \mathrm{Re} < 1, \\ \frac{24}{\mathrm{Re}^{0.6}} & \text{if } 1 < \mathrm{Re} < 800, \\ 0.44 & \text{if } \mathrm{Re} > 800. \end{cases} \tag{3}$$

The particles inside the disk are assumed to follow a Safronov initial mass function, $\xi(m) = n \exp(-m/\langle m \rangle)$ [12], where $\xi(m)$ is the number of particles per unit mass interval. This is conveniently represented by the mass generating function

$$m(X) = m_l - \langle m \rangle \ln\left[1 - \left(1 - e^{((m_l - m_u)/\langle m \rangle)}\right) X\right], \tag{4}$$

where m_l is the minimum mass, m_u is the maximum, $\langle m \rangle$ is the mean mass, X is a random variable uniformly distributed in the interval [0, 1], and $dX/dm \propto \xi(m)$ ($\langle m \rangle \geqq m_l$). The mass generating function has been computed in the way described, for example, in Kroupa, Gilmore, and Tout [13].

4.1. Protoplanetary Nebula Properties

Before solving the motion equations (1), we need to specify the temperature and surface density profiles of the basic unperturbed state. For the purpose of this work, we have adopted a simple formulation for the physical characteristics of the protoplanetary nebula. We take the total mass density $\sigma(r)$ and temperature $T(r)$ of the disk to be of the form

$$\sigma(r) = \sigma_o \left(\frac{r}{r_o}\right)^{-p}, \tag{5}$$

$$T(r) = T_o \left(\frac{L}{L_\odot}\right)^{q/2} \left(\frac{r}{r_o}\right)^{-q}, \tag{6}$$

where r_o is a reference radius with Keplerian orbital frequency Ω_o ($\sqrt{GM_o/r_o^3}$), temperature T_o, mass density σ_o, and velocity v_K, and L is the luminosity of the star in solar units. We assume that the disk is vertically isothermal at temperature $T(r)$, and we take $L = 1\ L_\odot$ throughout the paper.

The gas vertical scale height H is

$$H = c/\Omega = \sqrt{\frac{2R}{\mu}} T^{1/2} \Omega^{-1}. \tag{7}$$

The average gas density ρ_g is

$$\rho_g = \frac{\sigma}{2H} = \rho_o \left(\frac{r}{r_o}\right)^{-p+q/2-3/2}, \tag{8}$$

where $\rho_o = \sqrt{\mu/8RT_o}\sigma_o\Omega_o$. For the purpose of this paper, the midplane gas density is taken to be the same as this average value. The gas pressure is given by

$$P = \frac{\rho_g RT}{\mu}. \tag{9}$$

In this work we are interested in disks associated with T Tauri stars. As an example, for the circumstellar disk around T Tauri N [14] the temperature $T_{10\mathrm{AU}} = 26^{+34}_{-13}$ K, p is in the range 0.5–2.0, and q in the range 0.4–0.75. The probability that $p \geq 1.5$ is 65%. The physical parameters used in the rest of the paper are derived from the following standard model of the nebula: a thin disk in which the surface densities (both for gas and for particles) and the temperature are given by the decreasing power laws r^{-p} ($p = 3/2$) and r^{-q} ($q = 1/2$), respectively. These are plausible values according to the observational results pointed out above. At 1 AU from the central star, the surface densities are set to 17,000 kg m^{-2} for the gas (σ_g) and to 200 kg m^{-2} for the particles (σ_p), whereas the temperature is assumed equal to 280 K. The spatial density of the nebular gas at the equatorial plane is given by $\rho_g = 1.36 \times 10^{-6}\ (r/\mathrm{AU})^{-11/4}\ \mathrm{kg\,m^{-3}}$, with $\rho_g = \sigma_g/2H$. The density of the solid material is set to 2000 kg m^{-3}.

On the other hand, the small perturbation to the radial force due to pressure support is [16]

$$\begin{aligned} \psi &= -\left(\frac{1}{2\rho_g r\,\Omega^2}\right)\frac{\partial P}{\partial r} \\ &= \frac{(p+q/2+3/2)RT_o}{2r_o^2\Omega_o^2\mu}\left(\frac{r}{r_o}\right)^{(1-q)}. \end{aligned} \tag{10}$$

Therefore, the Cartesian components of the velocity of the gas, assuming initial unperturbed circular Keplerian flow, are now

$$\begin{aligned} V_X^g &= -(1-\psi)\Omega Y, \\ V_Y^g &= (1-\psi)\Omega X, \end{aligned} \tag{11}$$

where X, Y are the Cartesian coordinates in a frame of reference at rest and centered on the disk's host star. The reference surface mass density in terms of the global disk parameters is given by

$$\sigma_o = \left(\frac{2-p}{2\pi r_o^2}\right)\left(\frac{R_D}{r_o}\right)^{p-2} M_D, \tag{12}$$

where R_D and M_D are the radius and mass of the disk, respectively. If the protoplanetary disk has a certain mass, then another effect to include in the calculations is the self-gravity of the disk. If we consider a thin disk, the potential is given by

$$V(r) = -\frac{G}{r}\int_0^{2\pi}\int_{R_*}^{R_D}\frac{\sigma(r')r'dr'd\phi'}{\sqrt{1+\left(\frac{r'}{r}\right)^2 - 2\frac{r'}{r}\cos\phi'}}, \tag{13}$$

where R_* is the stellar radius. After some algebra, the radial force (assuming symmetry) is given by

$$\Psi = \Lambda\int_{R_*}^{R_D}\left[K(k) - \frac{1}{4}W\left(\frac{r'}{r} - \frac{r}{r'}\right)E(k)\right]k\sigma(r')\sqrt{r'}\,dr', \tag{14}$$

where $k^2 = 4rr'/(r+r')^2$, $W = k^2/(1-k^2)$, $\Lambda = G/r^{3/2}$, and

$$K(k) = \int_0^1 \frac{dx}{\sqrt{(1-x^2)(1-k^2x^2)}}, \tag{15}$$

$$E(k) = \int_0^1 \sqrt{\frac{1-k^2t^2}{1-t^2}}\, dt. \tag{16}$$

As a particular case, if we consider $\sigma(r) = \sigma_o/r^{3/2}$ the radial force is

$$\Psi = \frac{G\sigma_o}{r^{3/2}} \int_{R_*}^{R_D} \left[K(k) - \frac{1}{4} W \left(\frac{r'}{r} - \frac{r}{r'}\right) E(k)\right] kr'\, dr'. \tag{17}$$

This evaluation is very time-consuming and thus from a computational point of view it is better to fit the variation of the force across the disk by a power-law expression of the form $\Psi \propto r^{\gamma}$, where $\gamma \approx 2.42$. As for the gas, to calculate the angular velocity of the disk we need to take into account the force contributions from both the central star, mass M_o, and the disk itself. Evaluating the gravitational contribution of the disk using elliptic integrals is straightforward [17], but computationally time consuming and not well suited for parallelization. For computational convenience we adopt the approximation (for the star–disk system)

$$\Omega(r) = \left[\frac{GM(r)}{r^3}\right]^{1/2}, \tag{18}$$

where

$$M(r) = M_o + 2\pi \int_0^r \sigma\, r\, dr. \tag{19}$$

This is equivalent to treating the disk mass as being distributed spherically when the gravitational force is calculated [15]. In our case

$$M(r) = M_o + 2\pi\sigma_o \int_0^r r^{-3/2}\, r\, dr = M_o + 4\pi\sigma_o r^{1/2}. \tag{20}$$

After substituting (20) into (18) we can easily find an approximate value for the angular velocity of the disk.

4.2. Vortex Velocity Field

In principle, one can argue that differentially rotating fluid systems such as circumstellar disks can produce vorticial motions and hence vortices could be important at some level. However, the exact mechanism by which vorticity is produced is still unknown. The circulation (vorticity) of a flow with a velocity field is defined as

$$\omega = \nabla \times \mathbf{V_g}. \tag{21}$$

The Navier–Stokes equations of the flow (see, e.g., [18]) take the form

$$\frac{\partial \mathbf{V_g}}{\partial t} + (\mathbf{V_g}\nabla)\mathbf{V_g} = -\frac{1}{\rho_g}\nabla P + \frac{\eta_g}{\rho_g}\nabla^2\mathbf{V_g} - GM_\odot\frac{\mathbf{r}}{r^3}, \tag{22}$$

$$\nabla\mathbf{V_g} = 0, \tag{23}$$

where P is the gas pressure. Taking the curl of (22),

$$\frac{\partial \omega}{\partial t} + (\mathbf{V_g}\nabla)\omega = (\omega\nabla)\mathbf{V_g} - \nabla\frac{1}{\rho_g}\times\nabla P + \frac{\eta_g}{\rho_g}\nabla^2\omega. \tag{24}$$

The second term on the right-hand side of Eq. (24) is a source term for the vorticity. This term is nonzero if $P = P(\rho_g, T)$ (baroclinic flow) and it vanishes if $P = P(\rho_g)$ (barotropic flow). In general, vorticity can be generated by the nonalignment of $\nabla 1/\rho$ with ∇P (baroclinic instability). The last term on the right-hand side of Eq. (24) is responsible for the viscous dissipation of the vorticity. In the absence of dissipation (inviscid fluid, $\eta_g = 0$) and for a barotropic flow, the flux of vorticity across a material surface is a conserved quantity (Kelvin's circulation theorem; see, e.g., [18]). If the flow is not exactly barotropic (e.g., as a result of reprocessing of stellar photons), then vorticity can be generated directly.

Under the approximations considered in this work, a steady vorticial velocity field should be a solution of the incompressible, two-dimensional Euler equation and it should approach the standard Keplerian velocity field outside the vortex. Following [4], let us consider the motion of a solid particle near a vortex located at a distance r_o from the Sun. The motion of the dust particle in an inertial frame of reference (X, Y) centered on the Sun under the assumptions considered in this paper is given by Eq. (1). For convenience, let us consider an additional frame of reference centered at the vortex and rotating with constant angular velocity $\Omega_o = \sqrt{GM_\odot/r_o^3}$, (x, y). This noninertial, Cartesian frame of reference is such that the y axis points in the direction opposite to that of the Sun. In the inertial frame of reference, $X = r\cos\theta$, $Y = r\sin\theta$ and in terms of the variables x and y centered on a point (r_o, θ_o) (the vortex) rotating with the disk,

$$r = r_o + y, \tag{25}$$

$$x = r(\theta - \theta_o) = r(\theta - \Omega_o t). \tag{26}$$

Hence

$$X = (r_o + y)\cos\left(\theta_o + \frac{x}{r}\right), \tag{27}$$

$$Y = (r_o + y)\sin\left(\theta_o + \frac{x}{r}\right). \tag{28}$$

In terms of these variables the velocities can be written

$$\dot{X} = \dot{y}\cos\left(\theta_o + \frac{x}{r}\right) - \Omega_v(r_o + y)\sin\left(\theta_o + \frac{x}{r}\right), \tag{29}$$

$$\dot{Y} = \dot{y}\sin\left(\theta_o + \frac{x}{r}\right) + \Omega_v(r_o + y)\cos\left(\theta_o + \frac{x}{r}\right), \tag{30}$$

where $\Omega_v = \Omega_o + \dot{x}/r - x\dot{y}/r^2$. Using Eqs. (27) and (28), the expressions for the velocity become

$$\dot{X} = -\Omega_o Y + \frac{\dot{y}}{r} X - (\Omega_v - \Omega_o) Y, \tag{31}$$

$$\dot{Y} = \Omega_o X + \frac{\dot{y}}{r} Y + (\Omega_v - \Omega_o) X. \tag{32}$$

These equations must approach the standard expressions,

$$(\mathbf{V_g})_x = \dot{X} = -\Omega Y, \tag{33}$$

$$(\mathbf{V_g})_y = \dot{Y} = \Omega X, \tag{34}$$

at large distances from the vortex. They describe the Keplerian, axisymmetric velocity field of the unperturbed (by vortical turbulence) gas. Under the approximations considered in this work, a steady vorticial velocity field should be a solution of the incompressible, two-dimensional Euler equation as well as approach the standard Keplerian velocity field outside the vortex. In the rotating frame of reference, our choice of vorticial field is

$$\dot{x} = -\frac{3}{2}\Omega_o y - \frac{3}{f^2 - 1}\Omega_o y \exp\left[-\frac{(x^2 + hy^2)}{2R^2}\right], \tag{35}$$

$$\dot{y} = \frac{3}{2}\frac{1 + f^2}{f(f^2 - 1)}\Omega_o x \exp\left[-\frac{(x^2 + hy^2)}{2R^2}\right], \tag{36}$$

where $h = 2f/(1 + f^2)$ and f is a parameter of the model. As stated before, this velocity field approaches the standard behavior ($\dot{x} = -3/2\Omega_o y,\ \dot{y} = 0$) outside the vortex. Inside the vortex, the above equations approach

$$\dot{x} = -\frac{3}{2}\Omega_o y \frac{1 + f^2}{f^2 - 1}, \tag{37}$$

$$\dot{y} = \frac{3}{2}\Omega_o x \frac{1 + f^2}{f(f^2 - 1)}. \tag{38}$$

These produce streamlines which are Keplerian ellipses with aspect ratio $f = a/b$ (a, b are the semi-axes in the x and y directions respectively). If f approaches 1 we obtain a circular vortex, if $f \to \infty$ we have an infinitely elongated vortex.

If we take into account Eqs. (31), (32), (35), (36), the velocity of the gas around the vortex becomes

$$(\mathbf{V_g})_x = -\Omega_o Y\left(1 - \frac{3}{2}y/r_o\right) + \Omega_f\left(\frac{2}{h}xX + 2yY\right)K_e, \tag{39}$$

$$(\mathbf{V_g})_y = \Omega_o X\left(1 - \frac{3}{2}y/r_o\right) + \Omega_f\left(\frac{2}{h}xY - 2yX\right)K_e, \tag{40}$$

where $\Omega_f = (3/2)(\Omega_o/r)(1/f^2 - 1)$ and $K_e = \exp[-(x^2 + hy^2)/2R^2]$. To the first order, Eqs. (39) and (40) converge to the standard expressions (33) and (34) in the neighborhood of

the vortex. Besides, following Godon and Livio [19, 20], we assume that cyclonic vortices dissipate quickly, while anticyclonic vortices can survive in the flow for hundreds of orbits. When more than one vortex is present, the anticyclonic vortices interact and merge together to form larger vortices; hence our calculations are restricted to a single vortex going around the Sun at a distance r_o in a circular, Keplerian orbit. Godon and Livio [19] have shown that the amplitude of the vortex behaves like $A \propto e^{-t/\tau}$, where τ, the decay time, increases as the viscosity decreases. This occurs because the last term in Eq. (24) is responsible for the viscous dissipation of the vorticity. To simulate the dissipation of the vortex we consider a time-dependent vortex radius given by

$$R = R_o e^{-t/\tau}, \tag{41}$$

where R_o is the initial value of the radius of the vortex. Godon and Livio [19] have shown that the lifetimes of vortices are inversely proportional to the α parameter, 10–100 orbits with $\alpha \approx 10^{-4}$–10^{-3}. In our calculations the characteristic size (or radius) of the vortex is equal to the thickness of the nebula H and corresponds to the limit of subsonic motions.

In all the subsequent computations we scale lengths to the astronomical unit (the distance from the Earth to the Sun), times to the year, and masses to 10 kg, approximately the mass of a typical decimeter-sized particle.

5. THE PROGRAM

5.1. Integration Algorithm

Three different integration methods (Adams–Moulton, Runge–Kutta, and Bulirsch–Stoer) were tested to compare their performance and accuracy. The Adams–Moulton fourth-order predictor–corrector is a constant time-step integrator whose efficiency is low when the problem is stiff. In our case, this occurs when the particle starts to experience bulk perturbation. Therefore, variable time-step integrators, based on the Runge–Kutta and the Bulirsch–Stoer methods, were tested. For example, we used the fifth-order Cash–Karp Runge–Kutta integrator with an adaptative time step as described in Press *et al.* [21]. The Bulirsch–Stoer scheme, which is widely used in celestial mechanics because it includes simulations of the three-body problem [22, 23], proved to be the most appropriate and robust for our problem. The Bulirsch–Stoer method is the best-known way to obtain high-accuracy solutions to ordinary differential equations with minimal computational effort. The accuracy of the method was checked by integrating the system over a time scale of 10^5 yr, when the friction drag was omitted. For this time interval, the energy was found to be conserved to better than 10^{-9} and the angular momentum to better than 10^{-12}.

5.2. Parallel Issues

The need to break up a problem to run it on a multicomputer requires an approach different from that used in sequential programs. However, the extra programming effort required to use a multicomputer can yield commensurate speedups. The parallelization of an algorithm aims at attaining larger and larger N in the numerical representation of the real system; this is important not only to improve the spatial resolution, but also to get much more meaningful results, because a number of particles that is too low in comparison with the number of real bodies gives rise to (in our case) an unphysical contrast of the vortex against the disk.

In general, an efficient parallelization means a data distribution to the processors, the so-called domain decomposition (DD), that (1) distributes the numerical work as uniformly as possible and (2) minimizes the data exchange among the processors for distributed memory platforms like the Origin 2000. Moreover, such DD should be performed with a minimal computational cost. In the numerical simulation of collisionless protoplanetary disks it is relatively easy to deal with these tasks because of the noncollisional nature of the problem. Furthermore, for large N the average time step for all the CPUs is very similar, which causes a relatively homogeneous distribution of the workload (the number of calculations needed to evaluate the acceleration of a particle). For these reasons, a simple parallelization scheme was adopted, the master/slave approach. In summary, the particles start out on a master node, which then sets up the virtual machine, farms out the data and work, and manages the computational interaction among different CPUs. A copy of the disk quantities, including the vorticial velocity field, is passed to each CPU. The particles are divided equally among the CPUs, with the master getting any remaining particles. After a fixed time scale, the CPUs send the results to the master node to generate a global output and update the particle array. The use of the PVM message-passing library makes the code portable. The details of our algorithm in a PVM framework are now discussed.

First **NDISK2** calls **PVMFMYTID**( ) and **PVMFPARENT**( ). The **PVMFPARENT** call will return **PVMNOPARENT** if the task was not spawned by another PVM task. If this is the case, then **NDISK2** is the master and must spawn the other worker copies of **NDISK2**. **NDISK2** then reads the number of processes (CPUs) to use ($Ncpu$) and the number of particles to include in the calculations (N). Each spawned process receives $N/Ncpu$ particles. The particles are distributed at random to the CPUs, i.e., without any correlation with their spatial location. If $Ncpu$ does not divide N evenly, then the master performs the calculations on the remaining particles. Initial conditions are then generated and **NDISK2** spawns $Ncpu - 1$ copies of itself and sends each new task the data (arrays of positions, velocities, and masses) to perform the calculations. The message contains the lengths of the subarrays and the subarrays themselves. After the master spawns the worker processes and sends out the subvectors, the master then computes its part of the calculations. After a certain computing time scale the slave processes send their results to the master in order to update the arrays (erase the noncaptured particles and create new ones) as well as generate partial output and analyze application performance. The master process then receives the results from the slave processes. Our **PVMFRECV** call uses a wildcard (-1) for the task identification parameter. This indicates that a message from any task will satisfy the receiver. Using the wildcard in this manner results in a race condition. In this case the race condition does not cause a problem since it does not matter in which order we collect the partial arrays from the workers. This is appropriate for noncollisional systems; in a general case, unless one is certain that the race will not have an adverse effect on the program, race conditions should be avoided. Once the master receives all local subarrays it performs an analysis of the partial results generating some output (captured particles, averages of many interesting magnitudes, etc.) and checks for particles well inside the orbit of the vortex. These particles can no longer be captured by the vorticial structure; therefore they are eliminated from the calculations and replaced by new particles initially located at the outer edge of the ring. After the output is generated and the arrays are updated the master process transfers the new data to the worker processes to continue the calculations. The loop continues repeating until the integration time reaches the value of the total simulation time, an input variable. When the termination condition is achieved the master process finishes all the worker processes and stops itself.

A main problem with PVM-based parallel computing is that the standard software for batch queue processing available on SGI Origin 2000 computers is not able to checkpoint applications that use sockets; therefore, in case of hardware or software failure not caused by the application itself, automatic restart is not possible. Fortunately, we have been able to develop an internal checkpoint using the master process. Every time the master collects results from the slaves, it performs a common dump on a special recovery file. In this way, we can manually restart the simulation after a system failure without any further problems.

5.3. Performance

The good efficiency of the parallelization approach described above is basically shown by three facts: (1) behavior of the relative speedup close to the ideal behavior (linear in *Ncpu*), (2) rather negligible unbalancing of the workload, and (3) low parallelization overhead. The computational speed has been calculated using the same test run with different numbers of processors and we define the parallelization overhead as the CPU time needed by all the instructions which would not execute in a *sequential* (single CPU) run. To benchmark our code we use a relatively low particle number to minimize the impact on the other users of the server. However, our conclusions do not really depend on the particle number as long as the number of particles per CPU is greater than 100.

The relative CPU time for the same test run with 1200 particles is shown in Fig. 1. It was computed by normalizing the CPU time for a given simulation with *Ncpu* processors to the CPU time for a sequential run. For $N = 1200$, it appears to be rather good

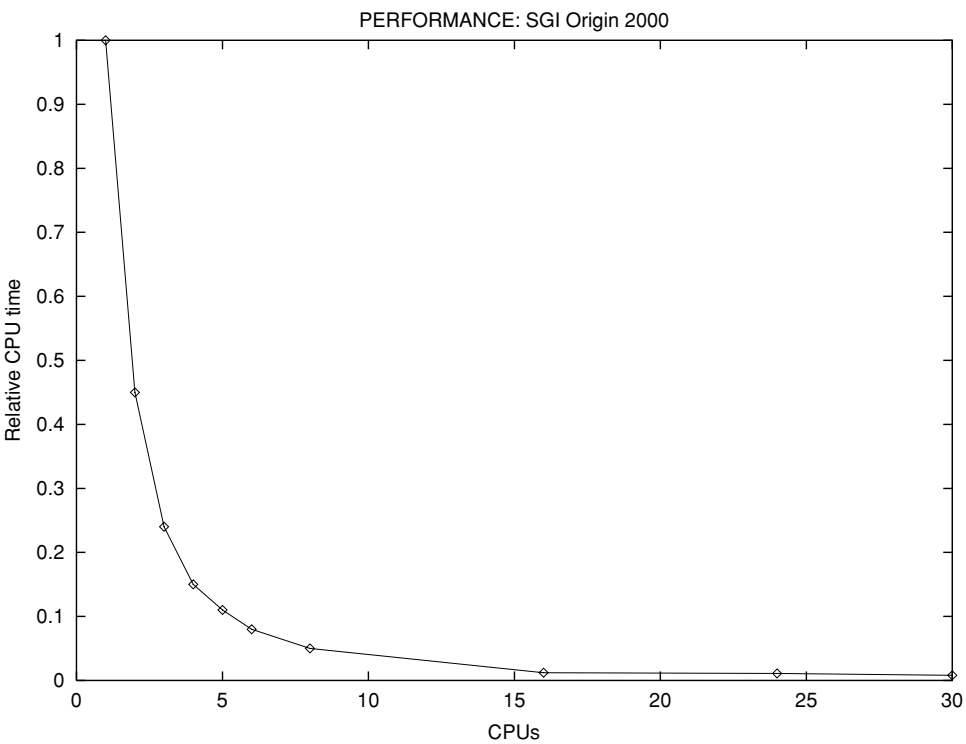

FIG. 1. Relative CPU time for the same test run with 1200 particles.

for $Ncpu \leq 6$ (i.e., $N/Ncpu \geq 200$). For more than 6 CPUs, the performance starts to degrade because the number of particles per CPU is too small. In principle, our calculations with 10^5 particles could use 10^3 CPUs (if available) without degrading the overall performance.

The computational workload is well balanced among the CPUs, as pointed out above. A natural way to quantify the load unbalancing is via the formula $l_u = (t_{\max} - t_{\min})/\langle t \rangle$, where $t_{\max}$ and $t_{\min}$ are, respectively, the maximum and the minimum CPU time spent by the workers to perform a block of calculations and $\langle t \rangle$ is the average for the workers. This load unbalancing is a time-dependent magnitude and it is always less than 15% for $N/Ncpu$ large enough (>40) and has an average value of about 5%.

The **NDISK2** code has been run on a massively parallel computer: the SGI Origin 2400 at the Centro de Supercomputación Complutense, Universidad Complutense de Madrid. This computer has 56 CPUs and 14 Gb of memory. However, the use of the PVM message-passing library makes the code portable to other massively parallel computers (like the Cray T3E) and clusters. Results from **NDISK2** have been checked against data generated by its predecessor, the sequential code **NDISK1** [24]. As expected, a small difference has been found because of the variation in floating-point roundoff errors.

5.4. MPI vs PVM

PVM uses mainly the concept of one program being the master and it then starts up *slave* processes on other nodes (hosts) that do all the work [25] (virtual machine concept). The slaves communicate back and forth with the master but rarely (never in our case) with each other. MPI on the other hand uses the concept of all processes running the same executable with the program doing different things depending on which process it is [25]. PVM can also utilize the SPMD (single program, multiple data) paradigm by using dynamic groups. PVM contains resource management, load balancing, and process control primitives but MPI is primarily concerned with messaging. PVM favors portability over performance and provides robust fault tolerance. MPI is more susceptible to faults and favors performance over flexibility. To compare both libraries we implemented another version of **NDISK2** that uses MPI instead of PVM. Although MPI's memory management seems to be slightly more efficient, the overall performance is lower. In general, and for the particular problem considered in this paper, the PVM version is about 1/3 faster than the equivalent MPI version. As expected, a small difference has been found in the final results from both versions because of the variation in floating-point roundoff errors. From a strictly computational standpoint, when a large difference in performance is found it should be understood. As pointed out before, the version of PVM running on SGI computers has been both optimized and improved. Libpvm provides a set of functions for packing types of data into messages and recovering it at the other end. Any primitive data type can be packed into a message in one of several encoding formats. The two most commonly used formats pack data into "raw" (host native) and "default" (XDR) formats. In our case the PvmDataRaw mode has been used. However, the difference in PVM and MPI performance is so large because our application is the only one using PVM and running on the SGI Origin 2000 of the Centro de Supercomputación Complutense. A large number of processes that use MPI cause unnecessary blocking and synchronization overheads, therefore the hardware environment favors PVM.

6. RESULTS

Finally, we discuss representative simulation results. Further applications of this model to large-scale turbulent structures in protoplanetary disks will be presented elsewhere. The simulation described in this paper has been used extensively to study dust capture by vortices. Color scale figures are available in the electronic version of the article only.

Figure 2 shows the instantaneous distribution of particles at four different times for a representative simulation. In this calculation we have used 36 CPUs on an Origin 2000 with PVM. It includes a realistic, exponentially decaying vortex with $f = 4$. The number of dust particles is $N = 400{,}000$, and they were distributed uniformly and randomly in the computational domain, 3.8–5.8 AU, inner/outer radius. The anticyclonic vortex stretches with time and follows a circular orbit with radius 5.2 AU. The top left panel shows the distribution of dust particles in the disk after 20 yr (about two orbital periods). The same

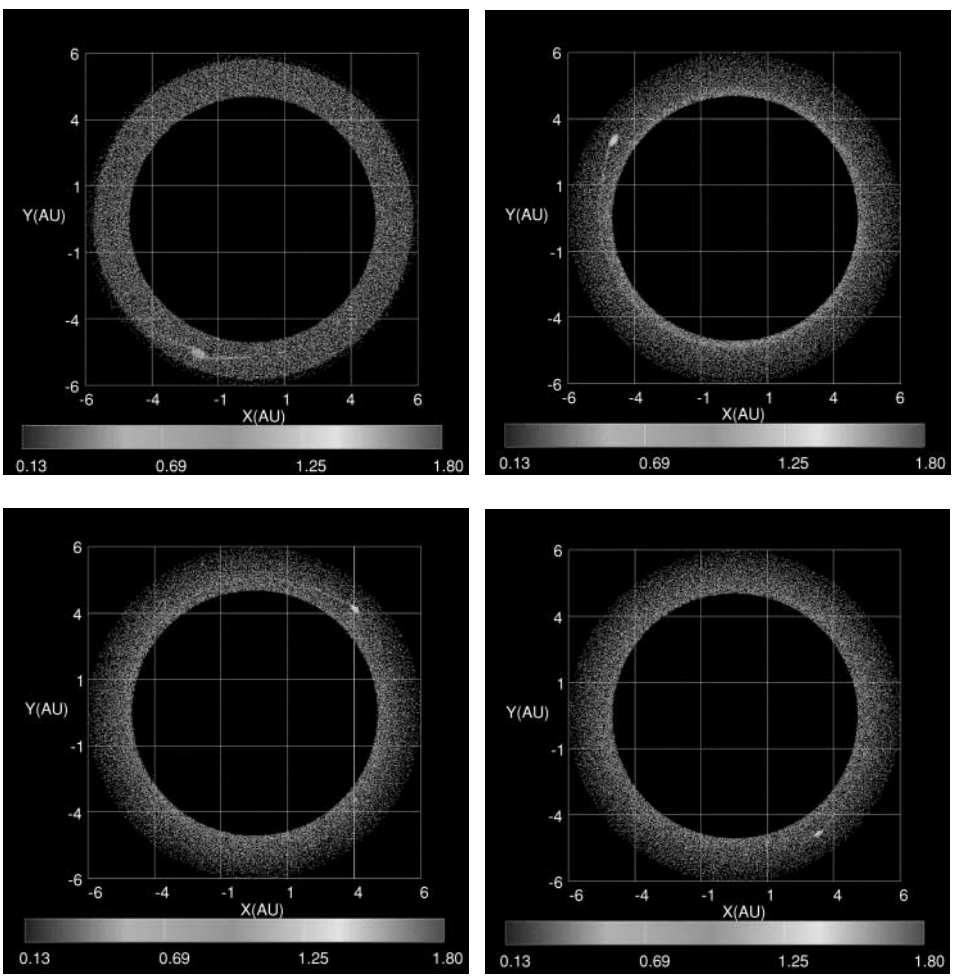

FIG. 2. Evolution of a protoplanetary disk that harbors a realistic vortex including decay and $f = 4$. The vortex is clearly visible through the sequence. The initial radius of the vortex is 0.37 AU. After 21 vortex periods the radius is 0.17 AU. The number of dust particles is 400,000, and they initially were distributed uniformly and randomly in the computational domain. Top left panel: 20 yr. Top right panel: 100 yr. Bottom left panel: 180 yr. Bottom right panel: 260 yr.

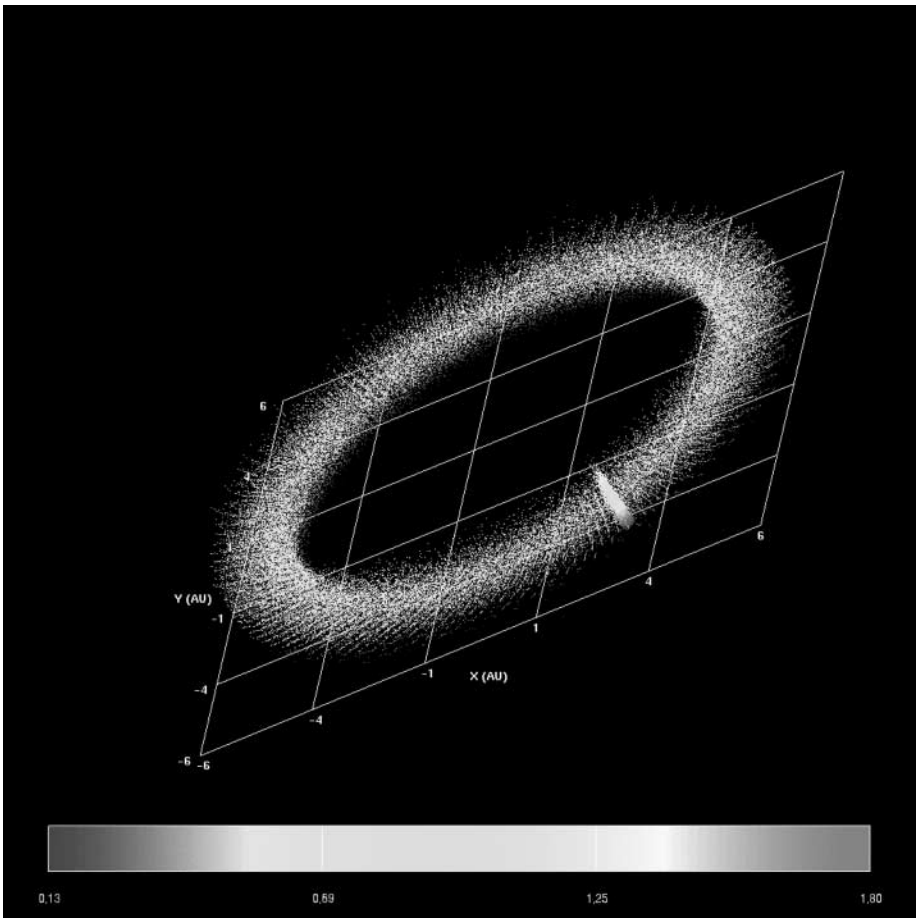

FIG. 3. Tilted view of the disk after 260 yr. The increase in particle density inside the vortex is clearly visible in the figure. This figure corresponds to Fig. 2, bottom right panel.

distribution after 100 yr is shown in the top right panel. The bottom left panel corresponds to 180 yr and the bottom right to 260 yr (about 25 orbits). It is rather clear from the sequence that starting from a uniform and random distribution in the disk, the dust particles concentrate inside the vortex with time. The efficiency of the capture-in-vortex process is a function of the ratio of the particle's response time against drag to its orbital period. After 25 orbits the size distribution inside the vortex is clearly different from the size spectrum in the disk. Figure 3 shows an artificially tilted view of the disk after 260 yr. For convenience, the size of the particles is represented as a third dimension. The local increase in particle density inside the vortex is clear from the figure. A detailed view of the vortex in the previous figures appears in Fig. 4, with the more massive captured particles preferentially located at the vortex core and the lightest particles more widely distributed. In our case, parallel computing with PVM allows for increased performance as well as particle number. Decreasing the cost of the simulation in terms of CPU time enables us to survey the free-parameter space of the model (f, particle size, disk mass, vortex location, etc.) in detail. On the other hand, increasing the particle number and using visualization tools like Advanced Visual System's AVS/Express make the interpretation of the results easier.

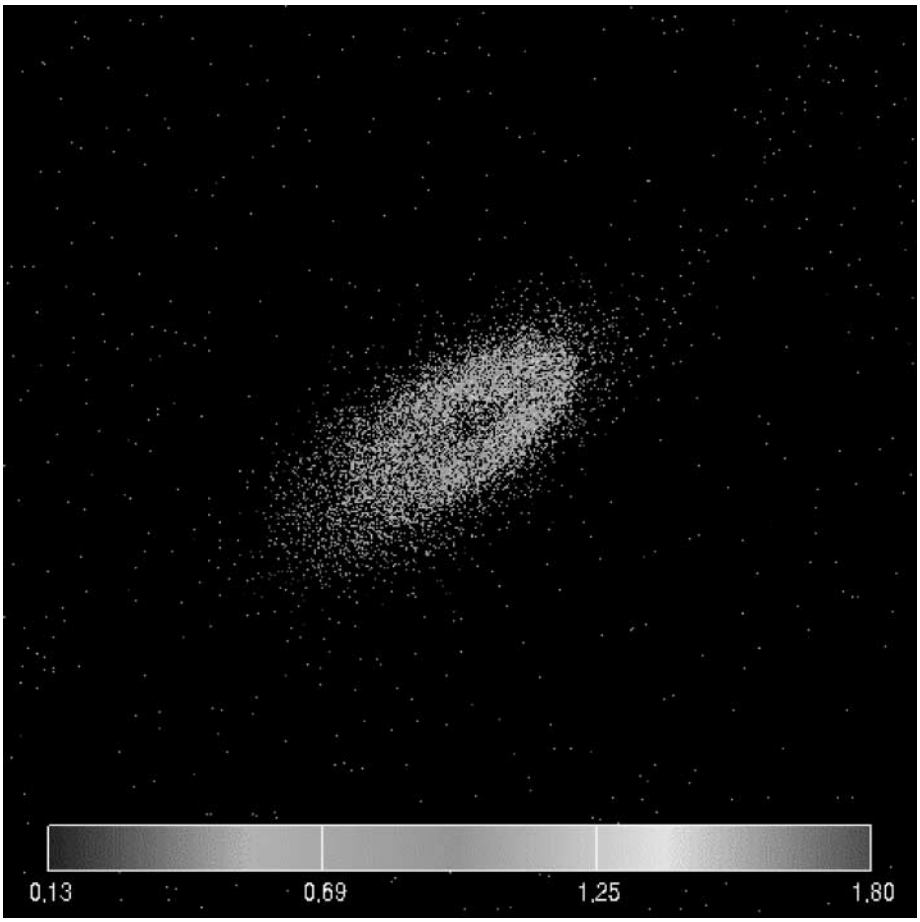

FIG. 4. Detailed view of the vortex in the disk after 260 yr. Stratification occurs inside the vortex with the smallest particles widely distributed and the larger particles more concentrated towards the vortex core (color scale is only available in the electronic version of the paper). This figure corresponds to Fig. 2, bottom right panel.

7. CONCLUSIONS

We have developed a parallel code for noncollisional simulation of the dynamical evolution of the solid material in a protoplanetary disk in which long-lived, large-scale vorticial structures perturb the motion of dust grains. Thus it has been possible to reduce CPU time and increase code performance. This enables us to run simulations with a large number of particles ($N \sim 10^5$–10^6) in nonprohibitive CPU times. NDISK2 is written in Fortran and designed to run on parallel computers using both the PVM and MPI message-passing libraries. For our particular problem—the dynamics of dust in a protoplanetary disk that includes a giant gaseous vortex—PVM is in general more efficient (up to 30% faster).

ACKNOWLEDGMENT

C.F.M. and R.F.M. thank the Department of Astrophysics of the Universidad Complutense of Madrid (UCM) for providing excellent computing facilities. The computations described in this paper were performed on the SGI

Origin 2000 of the Centro de Supercomputación Complutense (CSC) through the UCM project "Dinámica Estelar y Sistemas Planetarios" (CIP 454) under the supervision of Dr. M. Rego Fernández. We thank Rafael López and Luis Padilla from the CSC for advice and support during the completion of this work. To prepare this paper, we used the SIMBAD database operated at CDS, Strasbourg, France, the ASTRO-PH e-print server, and the NASA Astrophysics Data System.

REFERENCES

1. R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles* (Institute of Physics Publishing, Bristol/Philadelphia, 1988).
2. P. Barge and J. Sommeria, Did planet formation begin inside persistent gaseous vortices? *Astron. Astrophys.* **295**, L1 (1995).
3. P.-H. Chavanis, Trapping of dust by coherent vortices in the solar nebula, *Astron. Astrophys.* **356**, 1089 (2000).
4. C. de la Fuente Marcos and P. Barge, The effect of long-lived vortical circulation on the dynamics of dust particles in the mid-plane of a protoplanetary disc, *Mon. Not. R. Astron. Soc.* **323**, 601 (2001).
5. G. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine, A Users' Guide and Tutorial for Networked Parallel Computing* (MIT Press, Cambridge, MA, 1994).
6. W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message Passing Interface* (MIT Press, Cambridge, MA, 1994).
7. J. Mo, F. Romelfanger, R. J. Hanisch, D. C. Redding, S. Sirlin, and A. Boden, Implementation of an optical prescription retrieval code using PVM (parallel virtual machine) in a mixed architecture network, in *Astronomical Data Analysis Software and Systems V*, ASP Conference Series, edited by G. H. Jacoby and J. Barnes (1996), Vol. 101.
8. A. F. Boden, D. C. Redding, R. J. Hanisch, and J. Mo, Massively parallel spatially-variant maximum likelihood image restoration, in *Astronomical Data Analysis Software and Systems V*, ASP Conference Series, edited by G. H. Jacoby and J. Barnes (1996).
9. P. M. Ricker, S. Dodelson, and D. Q. Lamb, Cosmos: A hybrid N-body/hydrodynamics code for cosmological problems, *Astrophys. J.* **536**, 122 (2000).
10. H. R. Viturro and D. D. Carpintero, Parallelized tree-code for clusters of personal computers, *Astron. Astrophys. Suppl.* **142**, 157 (2000).
11. S. J. Weidenschilling, Aerodynamics of solid bodies in the solar nebula, *Mon. Not. R. Astron. Soc.* **180**, 57 (1977).
12. V. S. Safronov, *Evolution of the Protoplanetary Cloud and Formation of the Earth and the Planets* (Israel Program for Scientific Translations, Jerusalem, 1972).
13. P. Kroupa, G. Gilmore, and C. A. Tout, The effects of unresolved binary stars on the determination of the stellar mass function, *Mon. Not. R. Astron. Soc.* **251**, 293 (1991).
14. R. L. Akeson, D. W. Koerner, and E. L. N. Jensen, A circumstellar dust disk around T Tauri N: subarcsecond imaging at lambda = 3 millimeters, *Astrophys. J.* **505**, 358 (1998).
15. D. N. C. Lin and J. E. Pringle, The formation and initial evolution of protostellar disks, *Astrophys. J.* **358**, 515 (1990).
16. J. N. Cuzzi, A. R. Dobrovolskis, and J. M. Champney, Particle–gas dynamics in the midplane of a protoplanetary nebula, *Icarus* **106**, 102 (1993).
17. J. Binney and S. Tremaine, *Galactic Dynamics* (Princeton Univ. Press, Princeton, NU, 1987).
18. D. J. Acheson, *Elementary Fluid Dynamics* (Oxford Univ. Press, Oxford, 1990).
19. P. Godon and M. Livio, Vortices in protoplanetary disks, *Astrophys. J.* **523**, 350 (1999).
20. P. Godon and M. Livio, The formation and role of vortices in protoplanetary disks, *Astrophys. J.* **537**, 396 (2000).
21. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes* (Cambridge Univ. Press, Cambridge, UK, 1992).

22. R. Bulirsch and J. Stoer, Numerical treatment of ordinary differential equations by extrapolation methods, *Numer. Math.* **8**, 1 (1966).
23. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis* (Springer-Verlag, New York, 1980).
24. C. de la Fuente Marcos and R. de la Fuente Marcos, On the dynamics of dust grains in a hierarchical environment. I. Binary case, *Earth, Moon Planets* **81**, 145 (1998).
25. G. A. Geist, J. A. Kohl, and P. M. Papadopoulous, PVM and MPI: A comparison of features, *Calculateurs Paralleles* **8**, page (1996).