A cloudy Vlasov solution

C. Alard, S. Colombi

Mon. Not. R. Astron. Soc. **359,** 123–163 (2005) doi:10.1111/j.1365-2966.2005.08889.x

A cloudy Vlasov solution

C. Alard[1,2]★ and S. Colombi[1]★

[1]*Institut d'Astrophysique de Paris, 98bis boulevard Arago, 75014 Paris, France*
[2]*Observatoire de Paris, 77 avenue Denfert Rochereau, 75014 Paris, France*

Accepted 2005 January 26. Received 2005 January 26; in original form 2004 June 30

ABSTRACT

We integrate the Vlasov–Poisson equations giving the evolution of a dynamical system in phase space using a continuous set of local basis functions. In practice, the method decomposes the density in phase space into small smooth units having compact support. We call these small units 'clouds' and choose them to be Gaussians of elliptical support. Fortunately, the evolution of these clouds in the local potential has an analytical solution that can be used to evolve the whole system during a significant fraction of dynamical time. In the process, the clouds, initially round, change shape and become elongated. At some point, the system needs to be remapped on round clouds once again. This remapping can be performed optimally using a small number of Lucy iterations. The remapped solution can be evolved again with the cloud method, and the process can be iterated a large number of times without showing significant diffusion. Our numerical experiments show that it is possible to follow the two-dimensional phase-space distribution during a large number of dynamical times with excellent accuracy. The main limitation to this accuracy is the finite size of the clouds, which results in coarse-graining the structures smaller than the clouds and induces small aliasing effects at these scales. However, it is shown in this paper that this method is consistent with an adaptive refinement algorithm which allows one to track the evolution of the finer structure in phase space. It is also shown that the generalization of the cloud method to four-dimensional and six-dimensional phase space is quite natural.

Key words: gravitation – methods: numerical – galaxies: kinematics and dynamics – dark matter.

1 INTRODUCTION

By solving the Vlasov–Poisson equations, one can integrate in phase space the evolution of a self-gravitating, collisionless system of particles in the fluid limit. Assuming that the density of the system in phase space is given by the function $f(x, v, t)$, the Vlasov–Poisson system is, in the proper choice of units,

$$\frac{\partial f}{\partial t} + v.\nabla_x f - \nabla_x \phi.\nabla_v f = 0, \tag{1}$$

$$\Delta\phi = 2 \int f(x, v, t)\, \mathrm{d}v. \tag{2}$$

Due to the high dimensionality, $2D$, of phase space, where D is the dimension of the space, this problem is usually approached with the traditional N-body method, i.e. by approximating the distribution function by a discrete set of particles. However, with modern supercomputers, it now becomes possible to start envisaging a direct phase-space approach with $D = 2$ and $D = 3$. In this paper, we are thus interested in solving the Vlasov–Poisson equations directly in phase space. We consider a new implementation in one dimension, $D = 1$, but we shall discuss its extension to higher numbers of dimensions. We now review phase-space methods already studied in the past. After that, we give a sketch of our 'clouds' implementation and explain what is new compared to earlier work. At the end of this introduction, we shall detail the plan of our paper, which is mostly devoted to the actual technical details involved in the implementation of our method.

A fundamental property of the Vlasov equation is the Liouville theorem, which states that the phase-space distribution function is conserved along trajectories of matter elements in phase space:

$$f[x(t), v(t), t] = \text{constant}. \tag{3}$$

The first numerical methods used in astrophysics to solve the Vlasov–Poisson equations in phase space exploited this property directly, using the so-called water-bag model (DePackh 1962; Hohl & Feix 1967). The idea of the water-bag model is the following. If one assumes that the distribution function is constant within a patch in phase space, it is enough to follow dynamically the boundary of the patch. Numerical implementation of the water-bag model is therefore rather straightforward (Hohl & Feix 1967; Roberts &

★E-mail: alard@iap.fr (CA); colombi@iap.fr (SC)

Berk 1967; Cuperman, Harten & Lecar 1971; Janin 1971). Even though this isocontour method is quite efficient and accurate, it is in fact very costly. Indeed, the distribution function develops increasing filamentary details during evolution by the effects of rolling up in phase space due to differential orbital speeds. Therefore, it is in principle necessary to add more and more points to sample the boundary of the patches as time passes. This is one of the major weaknesses of the water-bag method, which is fine grained in essence, except for initial conditions.

Other approaches for solving the Vlasov–Poisson equation are grid based, and a large part of the technical developments come from plasma physics. One of the most famous numerical implementations, since it inspired much subsequent work, is the splitting algorithm of Cheng & Knorr (1976). The splitting scheme consists in exploiting the Liouville theorem in two steps, while evolving the system during a time-step Δt:

$$\begin{aligned} f^*(x, v) &= f(x - v\Delta t/2, v, t), \\ f^{**}(x, v) &= f^*(x, v + \nabla_x \phi \Delta t), \\ f(x, v, t + \Delta t) &= f^{**}(x - v\Delta t/2, v). \end{aligned} \tag{4}$$

In the method of Cheng & Knorr, the distribution function is interpolated on a grid using either Fourier methods or/and splines. It is semi-Lagrangian in the sense that, to compute the value of the distribution function at a grid site, test-particle trajectories are resolved backwards up to the previous time-step, where the interpolation is performed. If the original implementation of Cheng & Knorr is one dimensional, the generalization to higher numbers of dimensions is straightforward (e.g. Gagné & Shoucri 1977; see Sonnendrücker et al. 1999 for a more recent perspective). The method of Cheng & Knorr was first applied in astrophysics by Fujiwara (1981), Nishida et al. (1981) and Watanabe et al. (1981).

In principle the algorithm of Cheng & Knorr can be used as it is, even when the filamentation effects discussed above occur at resolution scale, although it has to be adapted, e.g. by using an appropriate interpolation procedure to guarantee positivity of the distribution function and mass conservation (e.g. Besse & Sonnendrücker 2003, for latest state-of-the-art developments). However, an elegant solution was proposed by Klimas (1987) to overcome the problem of filamentation. It consists in writing the exact equation of evolution of the coarse-grained distribution function in velocity space. For that, he assumes a Gaussian smoothing window. The modified Vlasov equation giving the evolution of the smoothed distribution function includes a new source term. This method was applied to a splitting algorithm using Fourier decomposition (Klimas & Farrell 1994).

Other grid-based methods include hydrodynamic advection schemes: the Lax–Wendroff integration method (Shlosman, Hoffman & Shaviv 1979) or other finite difference methods, using, for the interpolation of the fluxes, standard 'upwind' and total variation diminishing (TVD) algorithms, convenient to deal with filamentation, such as the van Leer limited scheme and the piecewise parabolic method (PPM), or other schemes such as flux-corrected transport, the flux balanced method (see Arber & Vann 2002, for a review and a comparison between these last four methods), or the positive and flux conservative method (Filbet, Sonnendrücker & Bertrand 2001). A finite element method was also proposed (Zaki, Gardner & Boyd 1988), but no further development in that direction was performed, probably because this method involves the inversion of coarse but large matrices, a very costly operation in six-dimensional phase space.

It is now worth mentioning two interesting special cases: the lattice method (e.g. Syer & Tremaine 1995), and the solver of Rasio, Shapiro & Teukolsky (1989). In the lattice method, the motion of elements of phase-space density is restricted to a set of discrete points: the time, positions and velocities of 'particles' are restricted to integer values, and forces are rounded to the nearest integer. Such an algorithm has the advantage of solving simplectic equations of motion, which guarantees conservation of geometrical properties of the system and, in the limit of infinite resolution, converges to the 'smooth' solution and naturally enforces the Liouville theorem. The second solver is the spherical code of Rasio et al. (1989, see also Merrall & Henriksen 2003), which works in the fully general case. The principle of this code is to take full advantage of the perfect knowledge of initial conditions: whenever $f(x, v, t)$ has to be determined at some point of space, e.g. to compute accurately the potential, a test particle is followed back in time to find its initial position and the value of f associated to it. The necessary sampling at each time-step is estimated by a self-adaptive quadrature routine. This code is therefore quite costly, since at each time-step a full set of backward trajectories has to be recomputed. It has, however, the advantage of following with very good accuracy all the details of the distribution function and it is probably the most accurate code of this kind available. It presents theoretically the same advantages as the water-bag method, but at variance with the latter, it is able to preserve the smoothness of the distribution function.

Finally, alternative ways of solving the Vlasov–Poisson equations consist in computing the moments of the phase-space density with respect to velocity space and position space, and writing partial differential equations for these moments up to some order, with some recipe to close the hierarchy (e.g. White 1986; Channell 1995).

A good numerical implementation of the Vlasov–Poisson equations should stick as close as possible to equation (3). In particular, it should, as a direct consequence of the Liouville theorem, preserve as much as possible the topology of the distribution function. More specifically, to render the algorithm TVD, and therefore stable, the following conditions should be satisfied.

(i) The critical point population, i.e. the number of maxima, minima and saddle points of various kinds should be preserved, and if this is not possible, should not increase with time. Since all Eulerian implementations use splitting in x and v, this condition reduces in the two-dimensional phase space to preservation of monotonicity in that case.

(ii) The height of the critical points should be preserved, or at worst, the height of local maxima should decrease and the height of local minima should increase. This in particular guarantees the positivity of the distribution function.

These conditions state that if the solution deviates from the true one, it should only become smoother. This is the essence of modern advection methods, which try to preserve the features of the distribution without adding spurious small-scale features, such as oscillations around regions with very high gradients. For instance, van Leer and PPM methods are TVD, at variance with most semi-Lagrangian methods. The higher the order of the scheme, the more accurate is the resolution of fine features of the distribution. However, to preserve the TVD nature of the system, the implementations are lower order in sharp transition regions [condition (i)] and in general nearby local extrema [condition (ii)]. As a result this can introduce significant diffusive effects. Clearly, we see that in this respect, grid-based methods are thus inferior to the water-bag method, which optimally fulfils conditions (i) and (ii), since it follows isocontours of the phase-space distribution function, and thus its topology, in essence. However, the number of sampling points increases with time in the water-bag model and a fair comparison with grid-based

methods should allow adaptive mesh refinement. Clearly, the very particular implementation of Rasio et al. (1989) is very successful in that sense.

The method we propose here is very different from all the work discussed above. The basic idea is to decompose the density in phase space into small local units which can be represented using a continuous function with compact support. The total density in phase space is the sum of the local functions, to have a fully analytical and continuous representation of the phase-space density of the system. We will call these small units 'clouds', which will be chosen here to be truncated Gaussians. The evolution of the distribution function will be followed by solving the Vlasov equation for each of the individual clouds, plunged in the global gravitational potential. We allow the clouds to change shape during run-time, i.e. to transform from functions with initial round support (with the appropriate choice of units) into functions with elliptic support. These clouds also move: their centre-of-mass position in phase space follows standard Lagrangian equations of motion, as test particles would. If the potential is locally quadratic, our elliptical shape approximation is exact, and the Vlasov equation can be solved analytically in that case. This means that as long as the clouds are small compared to the radius of curvature of the force, or in other words, to the scales of variation of the projected density, this approximation, accurate to second order in space, is very good.

At some point, the local quadratic condition is no longer satisfied, typically after a fraction of orbital time, and the analytical solution ceases to be a good approximation of the density of the cloud in phase space: the whole system has to be remapped on a new basis of round clouds that give a smooth description of the density in phase space. To resample the distribution function with a new set of clouds (including initial conditions), we use a Lucy or van Citter iterative method. This, combined with the fact that remaps are not very frequent, eliminates diffusion almost completely, provided that the resolution limit of the simulation has not been reached, i.e. as long as filamentation is not a problem. As a time integrator, we use a second-order predictor–corrector with slowly varying time-step. Our algorithm is thus fully second order in time and in space, and nearly simplectic, since in the case that the time-step is constant, it reduces to standard leap-frog (see, e.g. Hockney & Eastwood 1988).

As a matter of fact and as we shall see, our method deals very well with filamentation, as it naturally coarse grains the distribution function at small scales. It is quite close to finite element methods, except that the elements are of changing shape and that the sampling grid moves with time. It is also close in some sense to the water-bag method, but in a coarse-grained way, as by construction equation (3) is satisfied exactly as long as the potential is locally quadratic. The remap procedure is iterative and costly, and that is one of the drawbacks of the method, similarly to finite difference methods. However, we do not need to perform it too often (typically every 10–20 time-steps). The Lagrangian nature of the method and the reinterpolation scheme makes our method very weakly diffusive, but it is not TVD: aliasing effects can appear in regions with high curvature. However, for the numerical cases we studied, these effects are not critical. Positivity of the distribution is enforced with Lucy reconstruction, but not with van Citter. Note that our method can be used only in warm cases, i.e. in cases where the distribution function is smooth and has some width in velocity space. It is not appropriate for the cold case, e.g. to describe in phase space the formation of large-scale structure in standard cosmological models.

This paper is organized as follows. In Section 2, we present the theoretical background intrinsic to our method, i.e. study the dynamics of a phase-space cloud in a quadratic gravitational potential. We perform a full perturbative stability study, taking into account small deviations from quadratic behaviour. In Section 3, we discuss the actual numerical implementation. In Section 4, we test our code by performing simulations of a stationary solution, an initially Gaussian profile and an apodized water-bag (a top-hat in phase space). In Section 5, we compare the results obtained with our code with the standard N-body method. In Section 6, we propose an adaptive refinement procedure, which allows one to increase resolution where needed, if details of the distribution function need to be followed at smaller scales. Finally, Section 7 summarizes the results and discusses some perspectives for the method, in particular its extension to higher numbers of dimensions and the treatment of the cold case.

2 THE METHOD: CONCEPTS

2.1 One-dimensional equations

In this section we will show that for a quadratic potential, the Vlasov equation for a cloud has an analytical solution. Let $\phi(x, t)$ be the gravitational potential of the system. Locally the quadratic approximation will read:

$$\phi(x, t) = \alpha_0(t)\, x^2 + \alpha_1(t)\, x + \alpha_2(t). \tag{5}$$

By taking the following cloud equation it is possible to show that the Vlasov equation forms a closed system:

$$f(x, v, t) = G\left[\lambda_0(x, t) + \lambda_1(x, t)\, v + \lambda_2(t)\, v^2\right], \tag{6}$$

where G is any smooth function (with continuous derivatives). The functions $\lambda_i(x, t)$, $i = 0, 1, 2$, which determine the geometry of the cloud in phase space, can be obtained by solving the Vlasov equation:

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} - \frac{\partial \phi}{\partial x}\,\frac{\partial f}{\partial v} = 0. \tag{7}$$

It is interesting at this point to separate the general motion of the cloud from the evolution of its internal geometry. We will thus use Lagrangian coordinates, which are defined by ($x^* \equiv x - x_{\rm G}$, $v^* \equiv v - v_{\rm G}$), where ($x_{\rm G}$, $v_{\rm G}$) are the phase-space coordinates of the centre of gravity of the cloud. Assuming that the cloud phase-space density $f(x, v, t)$ become $f(x^*, v^*, t)$ in this referential defined by the Lagrangian coordinates, we can write:

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial x^*}v^* - 2\alpha_0 x^*\frac{\partial f}{\partial v^*} = 0, \tag{8}$$

$$f(x^*, v^*, t) = G\left[\Lambda_0(x^*, t) + \Lambda_1(x^*, t)\, v^* + \Lambda_2(t)\, v^{*2}\right], \tag{9}$$

$$\frac{{\rm d}^2 x_{\rm G}}{{\rm d}t^2} = -2\alpha_0(t) x_{\rm G} - \alpha_1(t). \tag{10}$$

By inserting equation (9) in equation (8), we obtain a quadratic polynomial in v^*, which has to cancel for any value of v^*. Hence, we obtain three equations:

$$\frac{\partial}{\partial t}\Lambda_0(x^*, t) - 2\,\Lambda_1(x^*, t)\,\alpha_0(t)\, x^* = 0, \tag{11}$$

$$\frac{\partial}{\partial t}\Lambda_1(x^*, t) + \frac{\partial}{\partial x^*}\Lambda_0(x^*, t) - 4\,\Lambda_2(t)\,\alpha_0(t)\, x^* = 0, \tag{12}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\Lambda_2(t) + \frac{\partial}{\partial x^*}\Lambda_1(x^*,t) = 0. \quad (13)$$

The general solution of equation (13) is

$$\Lambda_1(x^*,t) = -\frac{\mathrm{d}\Lambda_2}{\mathrm{d}t}x^* + \psi_1(t). \quad (14)$$

By inserting the solution for Λ_1 in equation (12) and solving for $\Lambda_0(x^*,t)$ we obtain:

$$\Lambda_0(x^*,t) = \frac{1}{2}\left[\frac{\mathrm{d}^2\Lambda_2}{\mathrm{d}t^2} + 4\,\Lambda_2(t)\alpha_0(t)\right]x^{*2} - \frac{\mathrm{d}\psi_1}{\mathrm{d}t}x^* + \psi_2(t), \quad (15)$$

and finally by substituting equations (14) and (15) into equation (11), we obtain a quadratic polynomial in x^*, which must be zero for any value of x^*. The zeroth order approximation of this polynomial gives $\psi_2 =$ constant. The first order approximation of this polynomial reads $\mathrm{d}^2\psi_1/\mathrm{d}t^2 = -2\alpha_0\psi_1$. However, in the referential of the centre of gravity, the first moments of f with respect to x^* and v^* should be zero. This implies that the form $\Lambda_0(x^*,t) + \Lambda_1(x^*,t)v^* + \Lambda_2(t)\,v^{*2}$, which we know now to be a polynomial of order 2 in x^* and v^* with time-dependent coefficients, does not have any term either in x^* and v^*: it is a quadratic form in x^* and v^*, implying $\psi_1(t) = 0$. The second order approximation of this polynomial is

$$\frac{1}{2}\frac{\mathrm{d}^3\Lambda_2}{\mathrm{d}t^3} + 4\alpha_0(t)\frac{\mathrm{d}\Lambda_2}{\mathrm{d}t} + 2\frac{\mathrm{d}\alpha_0}{\mathrm{d}t}\Lambda_2(t) = 0. \quad (16)$$

Equation (16) can be solved with respect to $\alpha_0(t)$:

$$\alpha_0(t)\,\Lambda_2(t)^2 = -\frac{1}{4}\int \frac{\mathrm{d}^3\Lambda_2}{\mathrm{d}t^3}\Lambda_2(t)\mathrm{d}t + C. \quad (17)$$

Since

$$\int \frac{\mathrm{d}^3\Lambda_2}{\mathrm{d}t^3}\Lambda_2(t)\mathrm{d}t = \frac{\mathrm{d}^2\Lambda_2}{\mathrm{d}t^2}\Lambda_2(t) - \frac{1}{2}\left(\frac{\mathrm{d}\Lambda_2}{\mathrm{d}t}\right)^2, \quad (18)$$

we can write

$$\frac{1}{4}\frac{\mathrm{d}^2\Lambda_2}{\mathrm{d}t^2}\Lambda_2(t) - \frac{1}{8}\left(\frac{\mathrm{d}\Lambda_2}{\mathrm{d}t}\right)^2 + \alpha_0(t)\,\Lambda_2(t)^2 = C. \quad (19)$$

It is possible to simplify this equation a little more by using the following substitution: $\Lambda_2(t) = w(t)^2$ (here we assume that $\Lambda_2(t)$, which corresponds to the velocity dispersion, is positive). We obtain an equation which is easier to use for numerical integration:

$$\frac{1}{2}\frac{\mathrm{d}^2w}{\mathrm{d}t^2} + w(t)\alpha_0(t) = \frac{C}{w(t)^3}. \quad (20)$$

We can now finally rewrite equation (9) as follows:

$$f(x^*,v^*,t) = G\left[\beta_0(t)\,x^{*2} + \beta_1(t)\,x^*v^* + \beta_2(t)\,v^{*2} + \beta_3\right], \quad (21)$$

with

$$\beta_0(t) = \frac{1}{2}\frac{\mathrm{d}^2\Lambda_2}{\mathrm{d}t^2} + 2\,\Lambda_2(t)\alpha_0(t), \quad (22)$$

$$\beta_1(t) = -\frac{\mathrm{d}\Lambda_2}{\mathrm{d}t}, \quad (23)$$

$$\beta_2(t) = \Lambda_2(t), \quad (24)$$

and β_3 can be set to zero without any loss of generality. We then see that equation (19) has a simple geometric interpretation, since it can now be rewritten

$$\beta_0(t)\,\beta_2(t) - \frac{1}{4}\beta_1(t)^2 = 2\,C, \quad (25)$$

which says that the area of the elliptical section in phase space defined by equation (21), proportional to

$$\bar{A} \equiv \pi A = \frac{\pi}{\sqrt{\beta_0(t)\,\beta_2(t) - \frac{1}{4}\beta_1(t)^2}}, \quad (26)$$

remains constant in time.

2.2 Dynamical properties of the clouds

In this section, we develop a number of simple analytical calculations which will help to implement the numerical simulations. These analytical models will also help us to understand the limits of our approach.

2.2.1 Small-amplitude oscillations

The evolution of the cloud is given by equation (20). In general, the solution of this highly non-linear equation can be approached only by numerical means. However, if we consider small-amplitude oscillations around an equilibrium position, it is possible to linearize the equations and find a simple analytical solution. We should consider a stationary equilibrium position with a quadratic potential given by $\alpha_0(t) = q_0$. In this case, the solution to equation (20) is:

$$w(t) = w_0 = \left(\frac{C}{q_0}\right)^{1/4}. \quad (27)$$

Note that due to the Poisson equation,

$$\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = 2\rho, \quad (28)$$

it is possible to relate the stationary potential q_0 to the mean density ρ_0 around the cloud:

$$q_0 = \rho_0. \quad (29)$$

Now, we introduce a time-dependent perturbation of this stationary regime:

$$\alpha_0(t) = q_0 + \epsilon q_1(t) \ \text{ and } \ w(t) = w_0 + \epsilon w_1(t). \quad (30)$$

Then equation (20) reads:

$$\frac{1}{2}\frac{\mathrm{d}^2w_1}{\mathrm{d}t^2} + 4q_0w_1(t) + w_0q_1(t) = 0. \quad (31)$$

A general solution to equation (20) can be obtained by using a Fourier transform. We define the following Fourier transforms:

$$\tilde{w}_1(\omega) = \frac{1}{\sqrt{2\pi}}\int w_1(t)\mathrm{e}^{\mathrm{i}\omega t}\,\mathrm{d}t$$

$$\tilde{q}_1(\omega) = \frac{1}{\sqrt{2\pi}}\int q_1(t)\mathrm{e}^{\mathrm{i}\omega t}\,\mathrm{d}t. \quad (32)$$

Using these definitions, the general solution of equation (20) is

$$\tilde{w}_1(\omega) = 2w_0\,\frac{\tilde{q}_1(\omega)}{\omega^2 - {\omega_0}^2} + k_0\,\delta(\omega-\omega_0) + k_1\,\delta(\omega+\omega_0), \quad (33)$$

where k_0 and k_1 are two arbitrary constants and

$$\omega_0 = 2\sqrt{2q_0}. \quad (34)$$

Equation (33) shows clearly that a resonance may occur between the potential and the cloud oscillations at the frequency ω_0. However for this resonance to occur effectively, it would be required that all the nearby clouds which contribute to the local density resonate also at this frequency. But, the resonant frequency of the nearby clouds can be the same only if the local density around these clouds is identical, which would require a constant density whatever the position. A resonance happens only in this case, and we will discuss this very special case later. The dynamical properties of the cloud are also dictated by the motion of their centre of gravity which is given by equation (10). The solution to equation (10) can be decomposed as the solution of the homogeneous equation and a given solution to the whole equation. The homogeneous equation reads:

$$\frac{\mathrm{d}^2 x_\mathrm{G}}{\mathrm{d}t^2} + 2\alpha_0(t) x_\mathrm{G}(t) = 0. \tag{35}$$

Setting $x_\mathrm{G}(t) = x_0 + \epsilon x_1(t)$ and noting that equation (35) is similar to equation (20), we find the general solution in the linearized regime:

$$\tilde{x}_1(\omega) = 2\bar{w}_0 \frac{\tilde{q}_1(\omega)}{\omega^2 - \bar{\omega}_0^2} + \bar{k}_0 \delta(\omega - \bar{\omega}_0) + \bar{k}_1 \delta(\omega + \bar{\omega}_0), \tag{36}$$

with a resonant frequency, $\bar{\omega}_0 = \omega_0/2$. Thus all the dynamical frequencies of the clouds are multiples of the fundamental frequency $\sqrt{2q_0}$, and thus, with the help of equation (29), we infer that the dynamical time of the cloud is:

$$T_\mathrm{cloud} \simeq \frac{\pi\sqrt{2}}{\sqrt{\rho_0}}. \tag{37}$$

Obviously this dynamical time has little meaning for a cloud experiencing large potential variations on a short time-scale. In this case the non-linearity is dominant and the linear approximation is not suitable. However, for most clouds in a given system, this approximation will give an estimate of the local time-scale.

2.2.2 *Resonant self-oscillations*

As noted in the previous section, resonances between the cloud motion and oscillations would require that the frequency ω_0 is constant. Using equations (29) and (34), we see that this condition ω_0= constant would require that the density is constant. The simplest example of a system having constant projected density is a line in phase space, with a density constant along the line. The oscillation frequency for any point on the line is the same whatever the position angle of the line in phase space. Thus such system can have self-oscillation, and out of this case there is no regime of self-oscillation of the cloud system. Note that this one-dimensional system is similar to the case of a sphere of constant density that has been shown to have self-pulsation in three dimensions.

2.2.3 *Deviation from the quadratic approximation*

The cloud approximation assumes that the potential at the scale of the cloud is quadratic. Even if the quadratic approximation is a good description of the local potential, when evolving the system, we expect that the errors to the quadratic approximation will accumulate, and that at some point the cloud model will deviate from the proper solution. Our numerical scheme should stop evolving the system before the deviation is large, and it is the purpose of this section to estimate the magnitude of this deviation as a function of the dynamical time describing the system evolution. We will consider an additional cubic term in the potential, and we will study the effect of this term in the linear perturbative regime. The new equation for the potential reads:

$$\phi(x^*, t) = \alpha_0(t) x^{*2} + \epsilon\gamma(t) x^{*3}. \tag{38}$$

We introduce the corresponding perturbation for the density in phase space:

$$f(x^*, v^*, t) = f_0(x^*, v^*, t) + \epsilon f_1(x^*, v^*, t). \tag{39}$$

Using the Vlasov equation (7), we obtain the following equations:

$$\frac{\partial f_0}{\partial t} + \frac{\partial f_0}{\partial x^*} v^* - 2\alpha_0 x^* \frac{\partial f_0}{\partial v^*} = 0, \tag{40}$$

$$\frac{\partial f_1}{\partial t} + \frac{\partial f_1}{\partial x^*} v^* - 2\alpha_0 x^* \frac{\partial f_1}{\partial v^*} - 3\gamma x^{*2} \frac{\partial f_0}{\partial v^*} = 0. \tag{41}$$

We already know the solution to equation (40) from Section 2.1. Since clouds having Gaussian profile are particularly interesting for numerical applications, we make the following choice in equation (9):

$$f_0(x^*, v^*, t) = \exp\left[\beta_0(t)\, x^{*2} + \beta_1(t)\, x^* v^* + \beta_2(t)\, v^{*2}\right]. \tag{42}$$

Now, it is possible to find a general solution to equation (41) by using the following functional for f_1:

$$\begin{aligned} f_1(x^*, v^*, t) = {} & f_0(x^*, v^*, t) \\ & \times \left[\eta_0(t) + \eta_1(t) x^{*3} + \eta_2(t) x^{*2} v^* \right. \\ & \left. + \eta_3(t) x^* v^{*2} + \eta_4(t) v^{*3}\right]. \end{aligned} \tag{43}$$

By combining equation (43) with equation (41), we obtain a polynomial in v^* of the order of 3. As the coefficients of this polynomial must be identically zero, we find the following system of differential equations:

$$\frac{\mathrm{d}\eta_4}{\mathrm{d}t} + \eta_3 = 0, \tag{44}$$

$$\frac{\mathrm{d}\eta_3}{\mathrm{d}t} + 2\eta_2 - 6\alpha\eta_4 = 0, \tag{45}$$

$$\frac{\mathrm{d}\eta_2}{\mathrm{d}t} - 6\gamma\beta_2 + 3\eta_1 - 4\alpha\eta_3 = 0, \tag{46}$$

$$\frac{\mathrm{d}\eta_1}{\mathrm{d}t} - 2\alpha\eta_2 - 3\gamma\beta_1 = 0. \tag{47}$$

2.2.4 *Effect of time-dependent non-quadratic terms: a practical case*

In practice the potential felt by a cloud moving in a given system can be quite different from quadratic. The Poisson equation (28) shows that the non-quadratic terms are related to local gradients in the density distribution. The accurate estimation of the local potential depends on the details of the density distribution, thus to quantify the effects we will have to adopt a given density distribution. However, in general, as the cloud moves, the system evolves and changes its potential, but this potential variation is slow and can be neglected for practical purposes. Let us assume that the density distribution is given by the family of profiles:

$$\rho(x) = q\left(\frac{x}{h}\right), \tag{48}$$

where h is a scaleheight.

Using the properties of this density distribution and the Poisson equation, it is easy to analyse the scale properties of the

cloud equation. The coefficients of the local potential α_0 and γ in equation (38) can be rewritten using the scaled variable $\bar{x}^*$:

$$\alpha_0 = \frac{1}{2}\frac{\mathrm{d}^2\phi}{\mathrm{d}x^{*2}} = \rho(x^*) = q(\bar{x}^*), \tag{49}$$

$$\epsilon\gamma = \frac{1}{3}\frac{\mathrm{d}\rho(x^*)}{\mathrm{d}x^*} = \frac{1}{3h}\frac{\mathrm{d}q(\bar{x}^*)}{\mathrm{d}\bar{x}^*}. \tag{50}$$

Equation (49) shows that α_0 is independent of the scale of the density distribution. As a consequence, equation (19) is also scale-independent, and thus the quadratic solution $\Lambda_2(t)$ is unaffected by the scale of the density distribution. But the behaviour of the coefficient γ is different. According to equation (50), γ is scale-dependent. It is possible to evaluate the effect of this scale-dependence by forming an equation for $\eta_4(t)$ similar to equation (19) for $\Lambda_2(t)$. By combining equations (44), (45), (46) and (47), it is possible to construct the following equation for $\eta_4(t)$:

$$-6\frac{\mathrm{d}\gamma}{\mathrm{d}t}\beta_2 - 15\,\gamma\frac{\mathrm{d}\beta_2}{\mathrm{d}t} + 3\frac{\mathrm{d}^2\alpha_0}{\mathrm{d}t^2}\eta_4$$

$$+10\frac{\mathrm{d}\alpha_0}{\mathrm{d}t}\frac{\mathrm{d}\eta_4}{\mathrm{d}t} + 10\,\alpha_0\frac{\mathrm{d}^2\eta_4}{\mathrm{d}t^2} + \frac{1}{2}\frac{\mathrm{d}^4\eta_4}{\mathrm{d}t^4} + 18\,\alpha_0^2\,\eta_4 = 0. \tag{51}$$

It is easy to notice that the scaling property of γ and the scale-invariance of α_0 and Λ_2 imply that η_4 scales like $1/h$, which means that the correction introduced by the cubic term is inversely proportional to scale. Thus, we may solve the equation for η_4 at a given scale and generalize the result to any scale using the former scaling rule. The solution for a fixed scale can be calculated using equations (49), (50) and (51).

For our numerical implementation, we will adopt a fixed Gaussian density distribution for the system, $q(x) = \exp(-x^2)$. Given initial conditions in phase space, the evolution of the cloud crossing this density distribution can be solved numerically by integrating our former system of equations. The motion of the centre of gravity of the cloud itself can be computed easily by estimating the motion of a point mass in the potential of the system. The orbit of the cloud is defined by its initial position x_0, and its initial velocity. We will study the case of zero initial velocity and variable position x_0. Initially the cloud will be a round Gaussian of velocity dispersion equal to 1. The numerical integration of these equations shows that the deviation from the quadratic approximation is maximal for an initial position close to $3h$ (see Fig. 1). The maximum deviation observed in the system is directly related to the requirement of performing a Lagrangian remap. Thus we should study the behaviour of the error to the quadratic approximation near the $3h$ initial position. The relevant plot is shown in Fig. 2. As discussed before, the curve showing the deviation due to the cubic term is generic. The deviation for another scalelength h of the density distribution can be obtained by rescaling this curve. Because the scaling is inversely proportional to h, the error in the quadratic approximation will be dominated by the crossing of the shorter-scale density structures.

3 THE METHOD: ALGORITHM

In this section, we study practical implementations of the method. We examined two approaches, one assuming constant resolution in phase space, which will correspond to what we call a 'cloud in mesh' (CM) code, and the other one allowing local refinement in phase space using adaptive refinement trees, which we therefore call tree-code. There are several issues to be addressed while implementing the method. We list them here in the same order as they will be treated below.

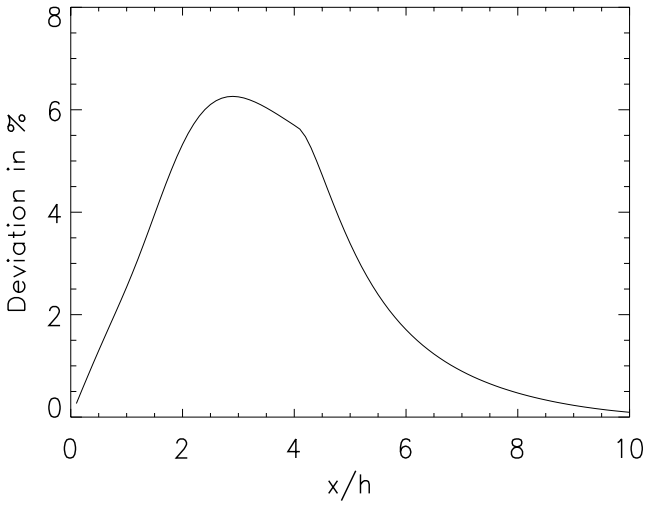

Figure 1. The maximum deviation from the quadratic approximation as a function of the initial position x_0. The distribution has a scalelength h 10 times larger than the cloud. The coordinate x_0 has been normalized by the scalelength of the distribution h. Note that the maximum deviation is at about $3h$ from the centre of the distribution.

(i) *Phase-space sampling of the distribution function* (Section 3.1). The question here is to decide how to chose our set of clouds such that it reproduces at best a given distribution function in phase space. This is necessary not only to set up initial conditions, but also to resample the distribution function during run-time with a new set of round clouds. Indeed, we know from our perturbative analysis (Section 2.2) that deviations from local quadratic behaviour of the potential increase with time, and that at some point the clouds will have the wrong axis ratio (be too elongated) and wrong orientation. To (re)sample the distribution function, we propose to use Gaussian clouds located on a (possibly adaptive) grid, with their masses estimated using either van Citter or Lucy deconvolution algorithms.

(ii) *Solving the Poisson equation* (Section 3.2). The issue here is to estimate accurately the forces exerted on each cloud as well as errors in their determination. These latter will indeed be used to

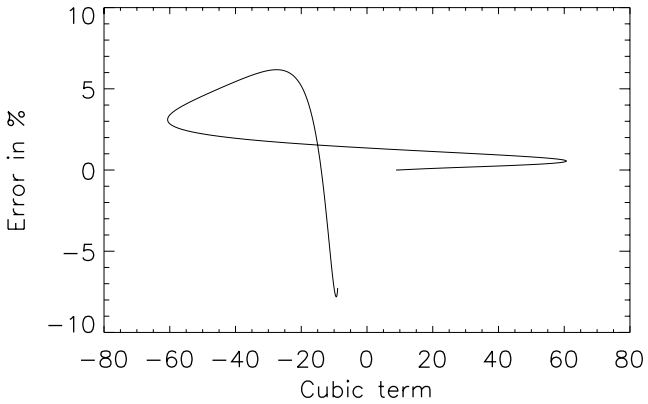

Figure 2. The shape of the deviation due to the cubic term as a function of the strength of the cubic term. We chose an initial position x_0 which maximizes the deviation from the quadratic approximation ($x_0 \simeq 3h$). Note that in case the distribution has a smaller scalelength the error must be rescaled accordingly. In practice, the minimum size of the distribution is about twice the cloud scalelength, for which the deviation can be five times larger.

quantify deviations from local quadratic behaviour of the potential. This is fairly easy in our one-dimensional problem, since the force exerted on a point of space is simply given by the difference between the mass at the right of the point and the mass at the left of the point. However, we will try here to experiment with methods which can be, in principle, easily generalized to higher numbers of dimensions without any serious increase in complexity.

(iii) *Run-time implementation and diagnostics* (Section 3.3). The choice of the time-step implementation is important. Here we propose a predictor–corrector of second order, which makes our method fully second order both in space and in time. We discuss how to compute the slowly varying time-step associated to this integrator. We also examine when a remap of the distribution with a new set of round clouds has to be performed, by using a criterion based on the cumulative errors on the forces due to non-local quadratic behaviour of the potential.

3.1 Phase-space sampling of the distribution function: practical implementation

When starting from given initial conditions, $f(x, v)$, it is necessary to set up an ensemble of round clouds that altogether reproduce, at best, f. During run-time, these clouds become more and more elongated and the approximation for a local quadratic potential breaks down: a remapping of the distribution function is needed with a new set of round clouds. The measured distribution function at this time becomes new initial conditions that, again, have to be sampled accurately. Here we explain in detail how we proceed to perform this remapping. In Section 3.1.1 we discuss the cloud shape and the mean cloud interspacing. Our choice is a Gaussian cloud of typical radius R, truncated at $4R$ and separated from its nearest neighbours by a distance $\sqrt{2}R$. In Section 3.1.2 we explain how we compute the cloud masses in order to reproduce at best the distribution function. The methods proposed are van Citter and Lucy deconvolutions algorithms. In Section 3.1.3, we propose a weighting scheme for filtering small-scale noise in the estimate of the distribution function obtained from the clouds. Finally Section 3.1.4 examines possibilities of enforcing conservation of basic quantities such as total momentum and total mass without adding significant diffusion. In addition to that, Appendix A details the algorithms used to compute the distribution function from the clouds. It shows how to speed up the calculation in the case when the clouds are round, thanks to the separability of the Gaussian window.

3.1.1 Choice of cloud shape and spacing

To sample the distribution function in phase space at a given time, we use an ensemble of clouds which are initially round (this can always be true if an appropriate normalization for positions and velocities is set). In this section, we assume that they are disposed on a rectangular grid of spacing Δ_g. The choice of our cloud shape is entirely determined by specifying the function G (equations 9 and 21), which should present the following features.

(i) It should have compact support. Indeed, we need the cloud to be the least extended as possible, since we use a local quadratic expansion of the potential within the cloud to compute the forces exerted on it.

(ii) It should be sufficiently smooth in order to resample a distribution function with continuous first and second derivatives (e.g. for the refinement procedure as discussed later). This is an essential feature of our method and a condition for it to perform well.

A good choice for the function G is a truncated exponential, which has the great advantage of being separable and makes the cloud Gaussian. The smoothness condition (ii) forces us to truncate this cloud rather far away from its centre. Our practical choice is a 4σ cut-off, $R_{\max} = 4\ R$.[1] This still induces small discontinuities of the order of 3×10^{-4}, as illustrated in the right-hand panel of Fig. 3. To minimize their effects we will use a weighted estimator to compute f, as detailed in Section 3.1.3 The remote nature of the cut-off has another disadvantage, which is one of the main drawbacks of our method, when it will be extended to higher numbers of dimensions: a large number of clouds contribute to the sampled distribution function at a given point of phase space, as illustrated by the lower right-hand panel of Fig. 3.

Finally, we have to determine the radius R (corresponding to a 1σ deviation) of our cloud as a function of grid spacing for best sampling of the distribution function. Basically, the choice of R determines by how much the resampled distribution function, $\tilde{f}$, will deviate from the true f. In order to sample smoothly the variations of f over the sampling grid, R/Δ_g should be of the order of unity. To minimize the cost of the sampling, it should be kept as small as possible. To find R, we estimate the quadratic error σ due to the representation of space by a finite number of Gaussian functions. As our approximation requires f to be nearly flat at the scale of the cloud, we will evaluate the residual in the case where the function to represent is constant, $f = 1$. For an infinite flat distribution σ reads

$$\sigma^2 = \frac{\int_{-\infty}^{\infty} [f(x, v) - \tilde{f}(x, v)]^2 \, \mathrm{d}x \, \mathrm{d}v}{\int_{-\infty}^{\infty} \mathrm{d}x \, \mathrm{d}v}. \tag{52}$$

For Gaussian clouds, f is represented by the functional:

$$\tilde{f}(x, v) = \frac{\Delta_g^2}{2\pi R^2} \sum_{i,j} \exp\left[-\frac{\left[(x - i\Delta_g)^2 + (v - j\Delta_g)^2\right]}{2R^2} \right]. \tag{53}$$

Note that the representation $f(x, v)$ can be rewritten as a convolution using Dirac series:

$$f(x, v) = \frac{\Delta_g^2}{2\pi R^2} \sum_{i,j} \int \exp\left[-\frac{\left(u_1^2 + u_2^2\right)}{2R^2} \right] \times \delta(u_1 + x - i\Delta_g)\delta(u_2 + v - j\Delta_g) \, \mathrm{d}u_1 \, \mathrm{d}u_2. \tag{54}$$

This rewriting as a convolution suggests that the equations should be analysed using Fourier transforms. As both the numerator and denominator in equation (52) are the norm of a function in real space, the transformation to Fourier space can be done easily by using Parseval's theorem. Using the symbol $\bar{f}$ to represent the Fourier transform of f, Parseval's theorem reads:

$$\int_{-\infty}^{\infty} [f(x, v) - \tilde{f}(x, v)]^2 \, \mathrm{d}x \, \mathrm{d}v = \int_{-\infty}^{\infty} |\bar{f}(k_x, k_v) - \bar{\tilde{f}}(k_x, k_v)|^2 \, \mathrm{d}k_x \, \mathrm{d}k_v, \tag{55}$$

and with the help of the convolution theorem:

$$\bar{f}(k_x, k_v) = \sum_{ij} \exp\left[-2\pi^2 R^2 \left(k_x^2 + k_v^2\right)\right] \times \delta(k_x - i/\Delta_g)\delta(k_v - j/\Delta_g). \tag{56}$$

Thanks to the rapid fall of the exponential, this sum is well approximated by a small number of terms. In particular a good

[1] More exactly $R_{\max} = 3.95\ R$, to avoid the circle of radius $R_{\max}$ intersecting any of the cloud centres.

approximation at the typical scale of interest is to consider only the terms (i, j) satisfying $|i| + |j| \leqslant 1$:

$$\int |\bar{f}(k_x, k_v) - \bar{\tilde{f}}(k_x, k_v)|^2 \, \mathrm{d}k_x \, \mathrm{d}k_v \simeq 4 \exp\left[-(2\pi R/\Delta_g)^2\right] \delta(0)^2. \tag{57}$$

The divergent factor $\delta(0)^2$ disappears once we apply the normalization by the denominator in equation (52). We thus finally find:

$$\sigma^2 \simeq 4 \exp\left[-(2\pi R/\Delta_g)^2\right]. \tag{58}$$

Our choice $R = \sqrt{2}/2$ corresponds to $\sigma \simeq 10^{-4}$, consistent with the discontinuities brought by our 4σ cut-off for the Gaussian. This is well illustrated in the lower left-hand panel of Fig. 3, which displays the deviations of f from unity for our choice of R/Δ_g. Experimentally, we find that their mean square is indeed equal to 10^{-4}, while the difference between local minima to local maxima is four times larger. Notice again that the truncation at $R_{\max}$ adds a source of noise at very small scales which has the bad property of presenting a significant skewness in its distribution. We shall see in Section 3.1.3 that with appropriate reinterpolation of the function f, it is possible to remove these sources of noise to a great extent, in fact totally in the case $f =$ constant considered here.

3.1.2 Choice of deconvolution scheme

Once the cloud shape is chosen as well as the size of the sampling grid, the problem of finding the cloud masses in order to sample correctly the distribution function remains open. To do that, an iterative procedure is necessary. We adopted in this paper two simple algorithms, van Citter and Lucy as we discuss now.

The sampled distribution function can be written

$$\tilde{f}(x, v) = \frac{1}{V} \sum_i M_i \, G\left(x - x_{\mathrm{G}}^i, v - v_{\mathrm{G}}^i\right), \tag{59}$$

where M_i is the mass of the cloud and the function $G(x, v)$ is normalized in such a way that its integral is V. The quantity V is given by $V = \Delta_x \Delta_v$, where Δ_x and Δ_v correspond to the grid site interspacing distance along the x- and v-coordinate, respectively. It is the area associated to each cloud so that $\int f(x, v) \, \mathrm{d}x \, \mathrm{d}v = \sum_i M_i$. Note that we used implicitly the following notation:

$$G(x, y) \equiv G\left(\beta_0 x^2 + \beta_1 xy + \beta_2 y^2\right). \tag{60}$$

We have to find M_i such that

$$\tilde{f}\left(x_{\mathrm{G}}^i, v_{\mathrm{G}}^i\right) \equiv \tilde{f}_i = f\left(x_{\mathrm{G}}^i, v_{\mathrm{G}}^i\right) \equiv f_i, \tag{61}$$

for each cloud position $(x_{\mathrm{G}}^i, v_{\mathrm{G}}^i)$ in phase space. A good first guess is simply

$$M_i^0 = f_i V. \tag{62}$$

However, with that choice of masses, $\tilde{f}$ will basically be equal to the convolution of f with the function G. Some deconvolution algorithm has to be applied in order to fit the function f better.

The van Citter algorithm is simply as follows. Given $\tilde{f}_i^n$ as computed at iteration n, then iteration $n + 1$ is

$$\begin{aligned} \delta_i^n &= f_i - \tilde{f}_i^n, \\ M_i^{n+1} &= M_i^n + \delta_i^n V. \end{aligned} \tag{63}$$

The algorithm is applied until some convergence criterion is fulfilled, $\max(|\delta_i^n|) \leqslant \delta_{\mathrm{c}}$. To maintain the domain of calculation as compact as possible during run-time, only values of f_i satisfying $f_i > \delta_{\min}$ are taken into account. In the simulations shown below, we take $\delta_{\min} = \delta_{\mathrm{c}}/2$, but $\delta_{\min}$ should in principle be kept as small as possible to have best conservation of the moments of the phase-space distribution function, e.g. energy. Similarly, after convergence, clouds contributing little to the reconstruction are depleted: one uses the reciprocal of equation (62) to estimate approximately their contribution, i.e. clouds with $|M_i/V| \leqslant \delta_{\min}$ are eliminated. This procedure has the defect of augmenting slightly aliasing effects in the neighbourhood of the regions where f cancels.

Given the sources of uncertainties associated with our choice of function G, δ_{c} should be large enough compared to $10^{-4} f_{\max}$, where $f_{\max}$ is the maximum value of f, since at this point we would capture spurious features and convergence would be made difficult due to the discontinuities brought by the cut-off. Experience shows that $\delta_{\mathrm{c}}/f_{\max}$ ranging from 0.0005 to 0.002 is a good compromise. Also, convergence might become difficult due to other aliasing effects. As well shall see later, the function f builds finer and finer structures with time, which cannot be reproduced correctly by our mapping when they appear at scales smaller than $\sim R$. Convergence is rendered very difficult in that case: one might choose therefore to stop iterating when n reaches some value, typically of the order of 10 according to our practical experiments. If convergence is not reached at this point, it is clear anyway that fine structures of the function f will not be reproduced correctly, even with a large number of iterations.

The major defect of the van Citter algorithm is that it does not guarantee positivity of the distribution function. An alternative to van Citter is the Lucy deconvolution algorithm:

$$\begin{aligned} \delta_i^n &= f_i/\tilde{f}_i^n - 1, \\ M_i^{n+1} &= M_i^n \sum_j \left(1 + \delta_j^n\right) G\left(x_{\mathrm{G}}^i - x_{\mathrm{G}}^j, v_{\mathrm{G}}^i - v_{\mathrm{G}}^j\right). \end{aligned} \tag{64}$$

Such an algorithm not only guarantees positivity, it is also in principle more accurate than van Citter, since the error δ_i^n is relative instead of absolute.[2] In practice, Lucy does better than van Citter, but is a little more unstable and is also more subject to aliasing effects. For our fixed-resolution simulations, we adopted the Lucy algorithm, but we used van Citter when testing refinement, as discussed later.

To illustrate the methods, Fig. 4 shows some results obtained when trying to resample the following distribution function, used later as initial conditions for some of our simulations: it is a top-hat apodized with a cosine:

$$\begin{aligned} f(x, v) &= \bar{\rho}, \quad x^2 + v^2 \leqslant \mathcal{R}^2, \\ f(x, v) &= \frac{1}{2}\bar{\rho}\left\{\cos\left[\frac{\pi}{2}(\sqrt{x^2 + v^2} - \mathcal{R})/\mathcal{R}_{\mathrm{apo}}\right] + 1\right\}, \\ & \qquad x^2 + v^2 \leqslant (\mathcal{R} + 2\mathcal{R}_{\mathrm{apo}})^2, \end{aligned} \tag{65}$$

where $\bar{\rho} = 1, \mathcal{R} = 0.7$ and $\mathcal{R}_{\mathrm{apo}} = 0.3$. The function $f(x, v)$ is shown and plotted in the top panels of the figure. The middle panels of the figure are the same but the difference $\delta f = \tilde{f} - f$ between the reconstructed distribution function and the true one is considered, when a small number of clouds is used, with interspacing $\Delta_g = \Delta_x = \Delta_v = 0.2$. We used 10 iterations in equations (63) and (64) and a cut-off value $\delta_{\min} = 0.5 \times 10^{-4}$. A larger number of iterations would not change significantly the level of convergence. Because Δ_g is of the same order as $\mathcal{R}_{\mathrm{apo}}$, aliasing effects are rather significant, particularly for Lucy where they reach a few per cent magnitude, because of the positivity constraint intrinsic to this

[2] In that case, δ_{c} is no longer expressed in terms of $f_{\max}$.

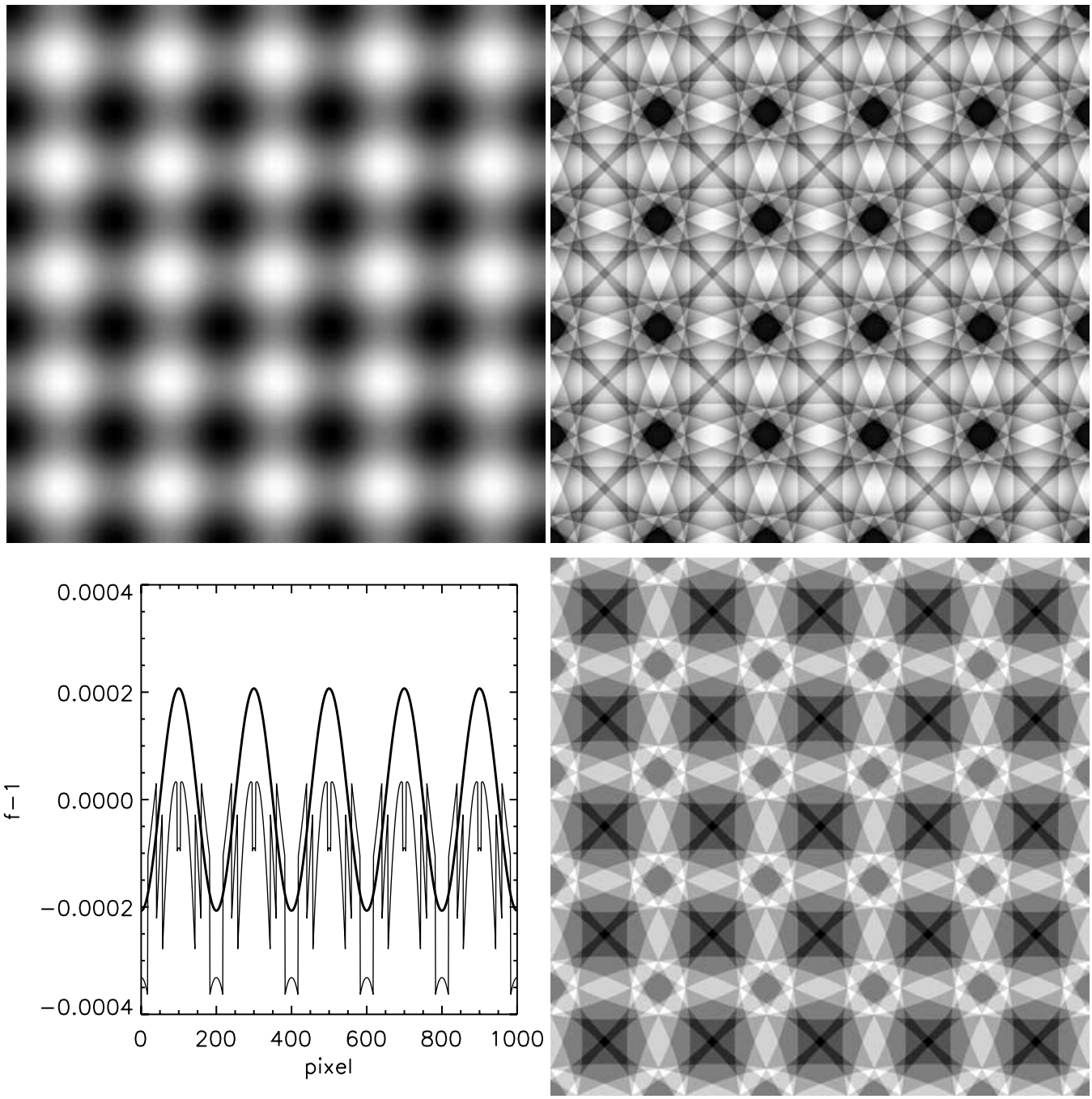

Figure 3. Sampling of $f(x, v)$= constant =1 by truncated Gaussians. We consider a set of 5×5 clouds in a periodic square of size $L = 5$. With these units, each Gaussian is of size $R = \sqrt{2}/2$ and is truncated at $R_{\max} = 3.95\ R$. The upper left-hand panel shows the variations of the sampled distribution function $\tilde{f}$ if the clouds had infinite extension (darker and lighter regions correspond to $\tilde{f} > 1$ and $\tilde{f} < 1$, respectively). They range with good accuracy from $2\sigma \simeq 2.1 \times 10^{-4}$ to -2σ (equation 58). The upper right-hand panel is the same, but the truncation is applied, which explains the discontinuities observed in $\tilde{f}$. In that case, fluctuations of $\tilde{f}$ are larger and are not symmetrically distributed, ranging approximately from -3.8×10^{-4} to 2.3×10^{-4}. The lower right-hand panel shows the cloud counts, i.e. for each point of space the number of cloud contributing to it. This count varies significantly, between 21 (dark) and 27 (light). Finally, the lower left-hand panel displays the values of $\tilde{f}(x, v = x)$ (the diagonal of the images) for the untruncated (thick smooth curve) and truncated case (thin irregular curve). The truncation adds a significant source of noise with a significant skewness in its distribution, which can become a source of systematic effects. This effect has to be minimized with the appropriate weighting scheme.

algorithm. The lower panels of the figure are the same as the middle panels, but a larger number of clouds was used, with interspacing $\Delta_g = 0.05$ now small compared to $\mathcal{R}_{\rm apo}$. As a result, aliasing effects are much less significant, of the order of $0.001 - 0.002$ and the difference between Lucy and van Citter has decreased. This shows that, to sample correctly variations of f over some length-scale ℓ, we need Δ_g to be sufficiently small compared to ℓ, typically $\ell/\Delta_g \sim$ a few units. Conversely, we see that in run-time the resampled f can be fully trusted only at coarse-graining scales of the order of a few Δ_gs.

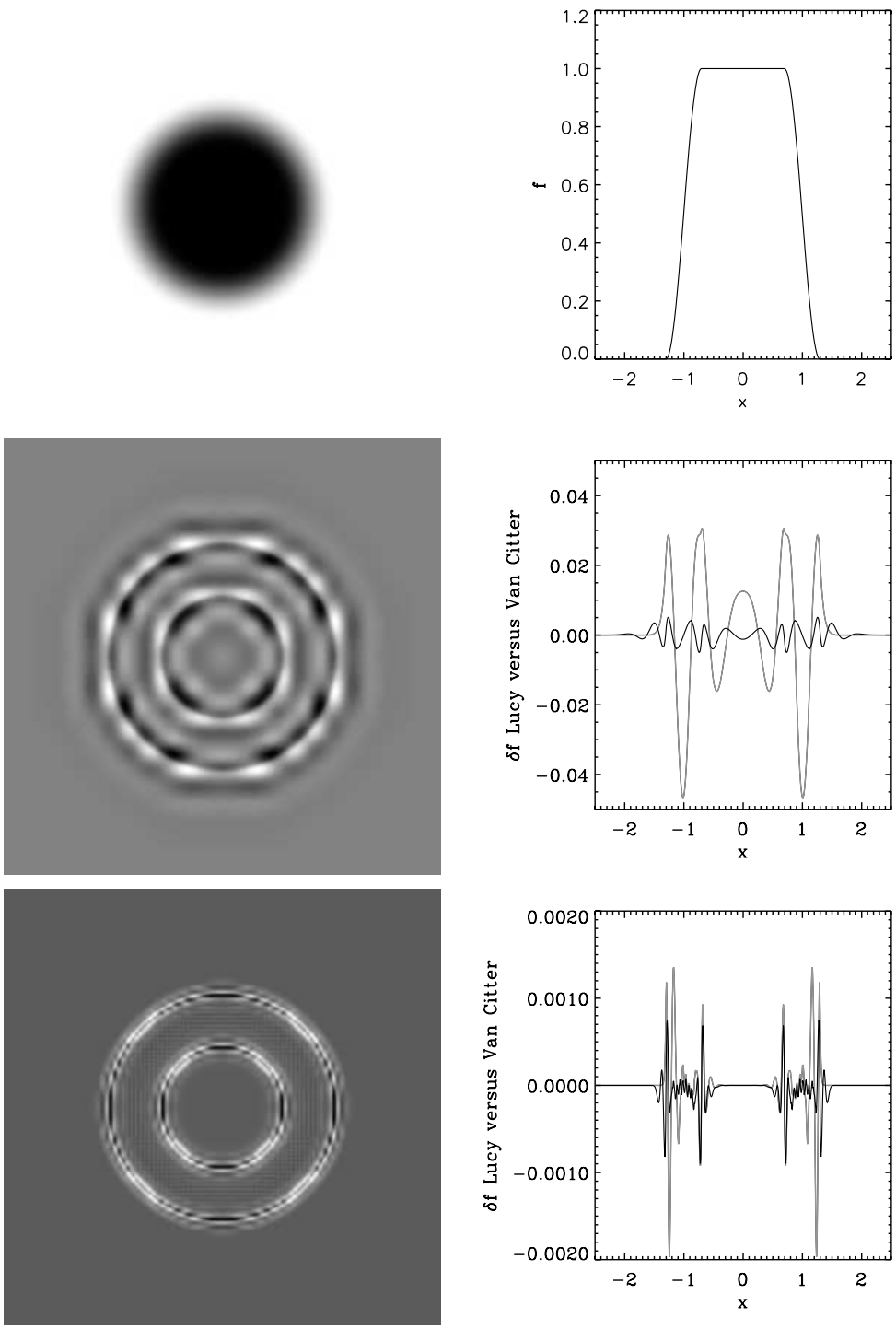

Figure 4. Comparison of the van Citter and Lucy algorithms for reconstructing a distribution in phase space. The upper left-hand panel displays the function $f(x, v)$ we aim to reproduce, as given by equation (65) [darker regions correspond to larger values of $f(x, v)$]. The upper right-hand panel shows $f(x, 0)$ as a function of x. The middle left-hand panel displays $\delta f \equiv \tilde{f} - f$, where $\tilde{f}$ is obtained by using the van Citter algorithm. A very similar result would be obtained for Lucy. The middle right-hand panel gives $\delta f(x, 0)$ as a function of x, both for van Citter (thin black curve) and Lucy (thick grey curve). To sample the distribution function, an interspacing of $\Delta_g = 0.2$ and a cut-off $R_{max} = 3.95\Delta_x$ was used. The lower left- and right-hand panels are the same but for higher resolution, $\Delta_g = 0.05$. Note that, for the four lowest panels, the weighting scheme discussed in Section 3.1.3 has been used to estimate $\tilde{f}$, following equation (66).

3.1.3 Filtering small-scale noise

After using either the van Citter or Lucy algorithms, one obtains from interpolation (59) a function $\tilde{f}$ which by definition reproduces at best the true values of f at the sampling points (x^i_G, v^i_G), given some convergence criteria. We just discussed problems of aliasing, which are intrinsic to the method and cannot be really avoided. They can be reduced only by using, e.g. refinement procedures discussed later,

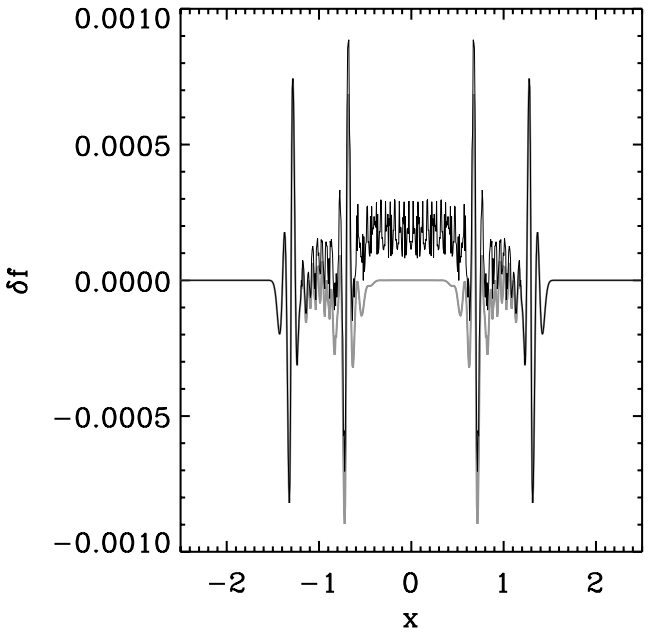

Figure 5. The difference $\delta f(x, 0)$ as a function of x, between the reconstructed (with van Citter) and true distribution function, whether equation (59) (thin black curve) or equation (66) is used to compute $\tilde{f}$. The function f to sample is the same as in Fig. 4 and the resolution is the same as in the lower panels of this figure.

or just by decreasing the cloud interspacing. However, it is possible to reduce the defects discussed in Section 3.1.1, namely small-scale variations due to the finiteness of phase-space sampling (upper left-hand panel of Fig. 3) and discontinuities due to the truncation of the clouds (upper right-hand panel of Fig. 3), to guarantee smoothness of the sampled distribution function. This is indeed necessary to avoid propagating this intrinsic noise during run-time, when a large number of reinterpolations is performed.

To minimize these defects, we perform a weighting different from equation (59):

$$f(x, v) = \frac{W}{V} \frac{\sum_i G\left(x - x_{\rm G}^i, v - v_{\rm G}^i\right) M_i}{\sum_i G\left(x - x_{\rm G}^i, v - v_{\rm G}^i\right)}, \tag{66}$$

where $W = \sum_{k,l} G(\Delta_x k, \Delta_v l) \simeq 1$ to ensure proper normalization. This weighting procedure is nearly equivalent to equation (59), since we expect the function $\sum_i G(x - x_{\rm G}^i, v - v_{\rm G}^i)$ to present small variations with (x, v) as shown by the upper right-hand panel of Fig. 3. However, it has the advantage of smoothing quite efficiently small-scale noise as illustrated by Fig. 5. For instance, in regions where $f =$ constant, it recovers exactly the true value of f within these regions (with a possible offset due to uncertainties in the convergence of the reconstruction). Hence, our weighting scheme would give exactly $f = 1$ everywhere in Fig. 3.

In what follows, when a new mapping with round clouds is necessary to resample an existing cloud distribution, we shall use equation (66) to compute f from the old set of clouds and equation (59) to compute $\tilde{f}$ from the new set of clouds in iterative procedures (63) and (64). We decided to do so in the last case to avoid introducing a bias in the new cloud-mass estimate.

3.1.4 Enforcing basic conservations

An intrinsic property of deconvolution methods such as Lucy's or van Citter's is that they conserve mass very well. However, we will have in practice to perform many remaps of the distribution function: any small but systematic deviation from mass conservation might have, on the long term, dramatic consequences. As a result, it might be necessary to enforce mass conservation, but this is not easy without adding diffusion. In order to reduce as much as possible this latter we proceed as follows. Suppose that the total mass of the system should be equal to M, and let $\tilde{M} = \sum_i M_i$ be the total mass obtained from the reconstructed clouds. If $M > \tilde{M}$, we compute the cumulative positive mass residues:

$$\Delta M_+ = \sum_{i, \delta_i^{\rm last} > 0} \delta M_i, \tag{67}$$

where $\delta M_i = \delta_i^{\rm last} V$ for the van Citter algorithm and $\delta M_i = M_i\, \delta_i^{\rm last}$ for the Lucy algorithm; these quantities correspond to the remaining uncertainty on the mass determination for each cloud after the last iteration, $n =$ last. Then, for each cloud having $\delta_i^{\rm last} > 0$, we increase its mass by a factor $\alpha\, \delta M_i$ with $\alpha = (M - \tilde{M})/\Delta M_+$. We proceed similarly if $M < \tilde{M}$, but by considering

$$\Delta M_- = \sum_{i, \delta_i^{\rm last} < 0} \delta M_i. \tag{68}$$

Our mass conservation scheme should minimize as much as possible diffusion effects, since the correction calculated for each cloud is proportional to the uncertainty on its mass determination. Furthermore, only clouds for which the mass should be increased (or decreased) in the direction of $M - \tilde{M}$ are modified. Note that if $\alpha > 1$, this means that our correction will tend to be above the residues. We can clearly fear for diffusion in that case.[3] Once total mass conservation is taken care of, we can pay attention to total momentum,

$$P_v \equiv \sum_i M_i v_{\rm G}^i, \tag{69}$$

which we always force to be zero by appropriate corrections of the velocities, if needed, since there is no external force exerted on the system we consider.

3.2 Solving the 1D Poisson equation

In this section, we explain in detail how we solve the Poisson equation. To do that, one needs first to estimate the local projected density, which is rather simple with our Gaussian clouds, as discussed in Section 3.2.1 Then one has to estimate the force exerted on each cloud, as well as its slope. To do that, we propose two methods: a tree-code approach based on decomposition of space on a binary tree (Section 3.2.2) and a 'cloud in mesh' (CM) approach based on sampling of space with a regular grid (Section 3.2.3).

3.2.1 Calculation of projected density

The projected density can be obtained for any value of x by summing all the individual contributions of the clouds. For a given Gaussian cloud, with elliptic basis and arbitrary orientation, integration over velocity space of equation (21) is in fact simple. With the help of equation (19), we obtain

$$\rho(x, t) = M \sqrt{\frac{2C}{\pi \Lambda_2(t)}}\, \exp\left(-\frac{2C}{\Lambda_2(t)}\, x^2\right) \tag{70}$$

[3] Also, it may be possible but rather unlikely that, e.g. $M > \tilde{M}$ and $\Delta M_+ = 0$. It is not worth enforcing mass conservation in this case, since the reconstruction itself is probably already very biased due, e.g. to strong aliasing effects.

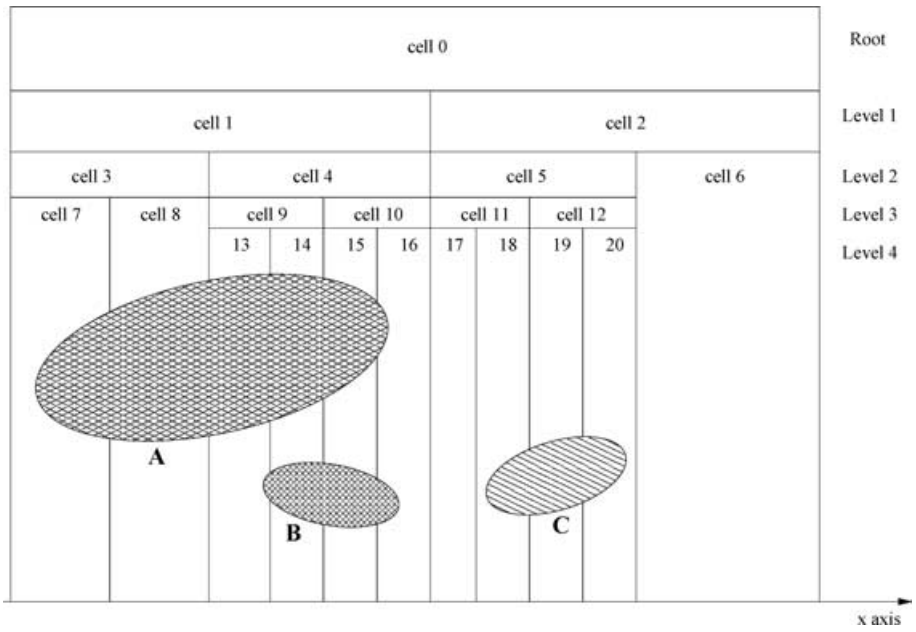

Figure 6. Sketch of our tree structure used to compute the forces within each cloud. Here, we consider the case $N_s = 3$, i.e. each cloud must intersect with at least three cells of the tree at a given refinement level. Note that for computing the forces for cloud A, only cells 7, 8, 9 and 10 of level 3 of refinement will be used, even if more refinement is available due to cloud B.

for a cloud of mass M. In this equation, we thus neglect the truncation of the cloud over velocity space, but this should not have any significant consequence. Indeed, in practice, to make sure that the total projected mass of each truncated cloud is equal to its true mass, we renormalize equation (70) by a factor very close to unity.

3.2.2 Practical implementation: a tree-code

Keeping in mind that we want to generalize the code to two and three dimensions, we experimented with an implementation which can be, in principle, easily extended to higher numbers of dimensions. The Poisson equation, in the appropriate units, is simply

$$\frac{\partial^2\phi}{\partial x^2} = 2\rho. \tag{71}$$

The approximation taken in this paper thus consists in simply assuming that $\rho=$ constant within a cloud. The force,[4] given by

$$-\frac{\partial\phi}{\partial x} = \int_{y\geqslant x}\rho(y,t)\,\mathrm{d}y - \int_{y\leqslant x}\rho(y,t)\,\mathrm{d}y = M_{\rm right} - M_{\rm left}, \tag{72}$$

is then locally fitted by a straight line within the cloud.

To estimate the force, we decompose the x-axis hierarchically on a binary tree (Fig. 6), by successively dividing segments into two parts of equal length, until each projected cloud intersects with at least N_s tree cells. Of course, there is an uncertainty on the real number of intersecting cells, $N_{\rm inter} \geqslant N_s$. It should be at least $N_s \geqslant 2$ and in fact large compared to that if we want to estimate errors on the forces. To compute the force within a cloud, given the number $N_{\rm inter}$ of tree cells intersecting with its projection on the x-axis, we store (i) the force F_i exerted on each cell centre, x_C^i, and (ii) a weight w_i proportional to $\rho(x_C^i - x_G) = \rho(x_C^{*\,i})$ as given by equation (70).

[4] In this paper, we indistinctly liken the quantity $-\nabla\phi$ to a force or an acceleration.

This weight is of course more important when the cell centre is close to the projected cloud centre. Then, given a list of (x_i, F_i), $i = 1, \ldots, N_{\rm inter}$, we use a simple weighted least-squares fit to adjust the force by a local straight line $F_{\rm fit}(x) = -2\alpha_0 x - \alpha_1$ (see equation 5). Our estimator for the error on the force then reads

$$\left(\frac{\Delta F}{F_{\rm max}}\right)^2 = \frac{1}{M^2\sum_{i=1}^{N_{\rm inter}} w_i}\sum_{i=1}^{N_{\rm inter}} w_i\left[F_i - F_{\rm fit}\left(x_C^i\right)\right]^2. \tag{73}$$

This weighted quantity roughly quantifies the deviation of the force from a straight line within the cloud. It is renormalized by the maximum force, $F_{\rm max}$, which in the units chosen here is equal to the total mass M of the system. The error on the force is indirectly related to the projected size of the cloud along the x-axis. If the cloud becomes considerably elongated, the error on the force will become larger in a fixed potential. Therefore, one might use $\Delta F/F_{\rm max} \leqslant$ some small value as a criterion to decide whether a remap of the distribution function with a new set of round clouds is necessary. However, as we shall discuss in more detail in Section 3.3.3, what really counts is the cumulative error,

$$E_{\rm cum} \equiv \int \Delta F(t)\,\mathrm{d}t. \tag{74}$$

To make sure that the force is estimated consistently the same way for all clouds (in particular to preserve approximately local smoothness properties), we stop refining the tree locally as soon as $N_{\rm inter} \geqslant N_s$, even if the tree has many branches due to much smaller clouds at the same location x, as shown by Fig. 6. Since we chose here $R_{\rm max} = 4R$ for the truncation of the cloud, one might think that $N_s = 8$ should be taken, to have at least two sampling cells per typical length-scale, R, of the clouds. However, we noticed that $N_s = 2$ is enough to make a very accurate determination of the force with our choice of $R/\Delta_x = \sqrt{2}/2$. This is illustrated by Fig. 7. To estimate the instantaneous error on the force, a larger value of N_s is necessary, typically $N_s \gtrsim 8$, while $N_s \gtrsim 4$ is enough to compute the cumulative error as used in Section 3.3.3.

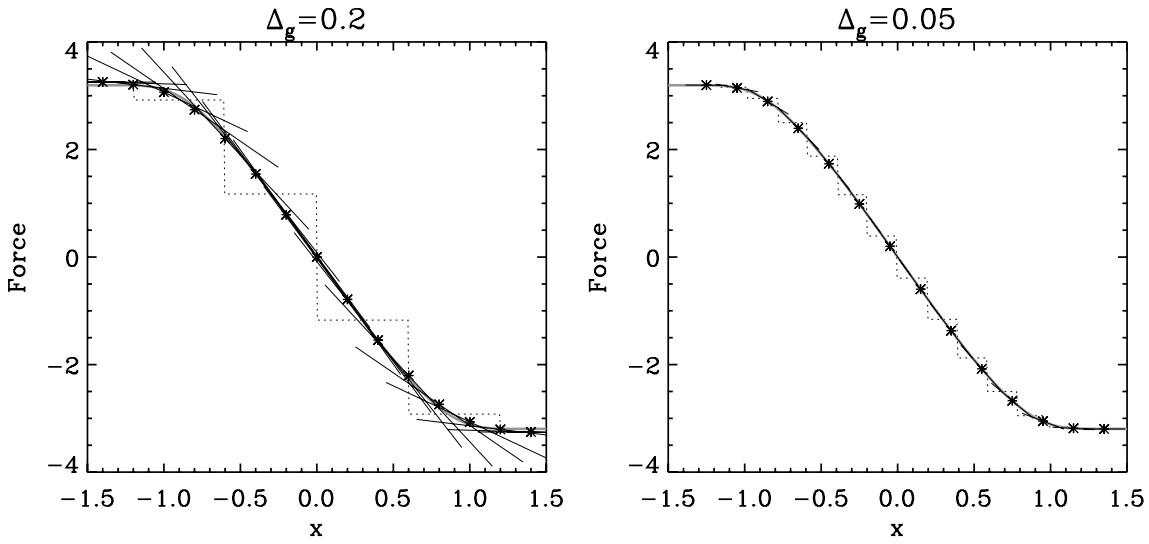

Figure 7. The calculated force for the examples considered in Fig. 4. The left- and right-hand panels correspond to the case $\Delta_g = 0.2$ and $\Delta_g = 0.05$, respectively. On each panel, there is a smooth thick grey curve: it corresponds to the exact force derived directly from integrating equation (65). There is a thin step-like dotted curve: it corresponds to the force computed for each cell of the binary tree at the resolution needed to sample each cloud with a least N_s cells. Here we took $N_s = 2$, and in practice we obtain $N_{inter} = 2$ or 3. There are stars nearly perfectly fitting the smooth thick grey curve: they give the force calculated at the centre of each selected cloud for doing the figure (a subset of all the clouds). A segment passes through each of these stars: it gives our weighted least-squares fit $F_{fit}(x) = -2\,\alpha_0 x - \alpha_1$, which trivially reduces to the line passing through the two sampling points when $N_{inter} = 2$. It is interesting to note that the estimate we obtain for the force at the cloud centres is very accurate, although one might notice a slight offset between symbols and the thick smooth grey line in the left-hand panel. The projected length of each segment on the x-axis is equal to the projected length of each cloud (up to cut-off). As we see, significant deviations from our local quadratic assumption are expected for the potential in the left-hand panel. These deviations are much less visible on the right-hand panel. The typical weighted deviation calculated from equation (73) by using $N_s = 8$ gives $\Delta F/F \simeq 0.03$ and 0.002 for left- and right-hand panels, respectively.

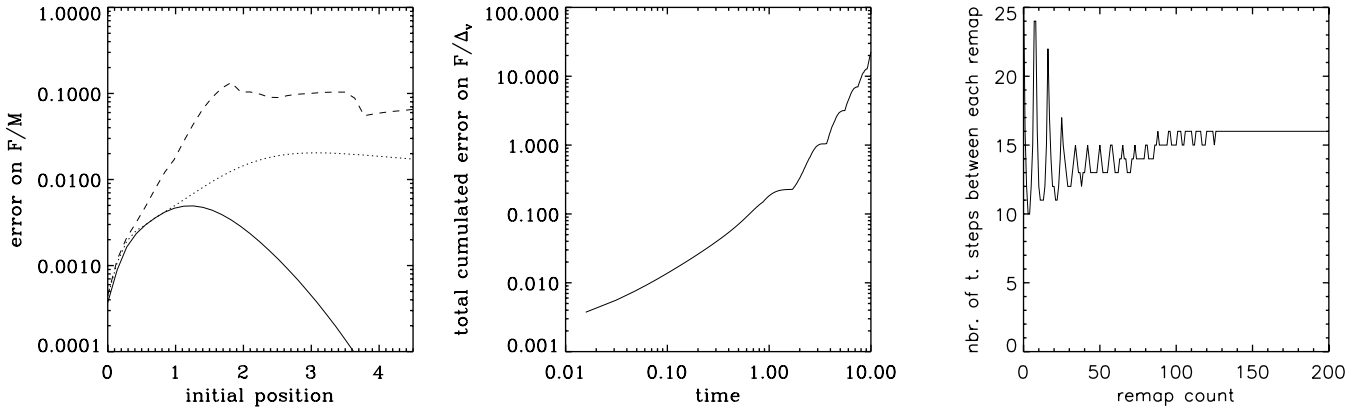

Figure 8. Left-hand panel: the maximum instantaneous error, $\Delta F/F_{max}$, in the force calculation (equation 73) as a function of initial cloud position, measured in a simulation without remapping. The initial conditions considered here are a truncated Gaussian of size $\mathcal{R}$ as given by equation (84). The initial position, x_{ini}, of the cloud is expressed in units of h. To reproduce a case similar to the one studied in Fig. 1, the cloud size R is given by $R/h = 0.1$. Three dynamical times are considered, $t = 0.5$ (solid), 1.5 (dots) and 6.0 (dashes). It is difficult to make an exact comparison to Fig. 1, given that fact that our estimate of $\Delta F/F_{max}$ is a quadratic dispersion. However, the general order of magnitude found on the error agrees with the results of Section 2.2.4. The dotted curve which corresponds approximately, dynamically, to the parameters chosen to make Fig. 1, agrees well, at least qualitatively with Fig. 1. Middle panel: the maximum cumulative error, $E^{max}_{cum/\Delta v}$, as a function of time. A remapping should be performed each time this error exceeds some threshold. Right-hand panel: the number of time-steps, n_{remap}, between each remap as a function of remap count, when the condition (85) is fulfilled, with $E_{max} = 0.05$. The simulation used to make this plot is exactly the same as the one used for the left-hand and middle panels, except that Lagrange remap was enforced. The simulation was stopped after 3000 time-steps, at $t \simeq 39$, corresponding to 35 dynamical times according to equation (37) with $\rho_0 = \bar{\rho}$. Note that the variations of n_{remap} decrease with time: n_{remap} seems to converge progressively to some fixed value, as expected, while the system relaxes to a stationary state.

3.2.3 Alternative practical implementation: 'CM' code

The results presented in this paper use a tree-code. The tree-code has the advantage of being rather flexible and easily generalizable to higher numbers of dimensions. It also represents a natural ground for adaptive refinement in phase space, as we shall see later. To simplify the approach, it is also possible to assume fixed resolution in space, which allows one to use a fast Fourier transform (FFT) or relaxations methods to solve the Poisson equation, similarly as in particle-in-mesh codes (PM). However, at variance with PM, a much more accurate 'cloud-in-mesh' interpolation (CM) is performed. In one dimensional case, the resolution of the Poisson equation is rather

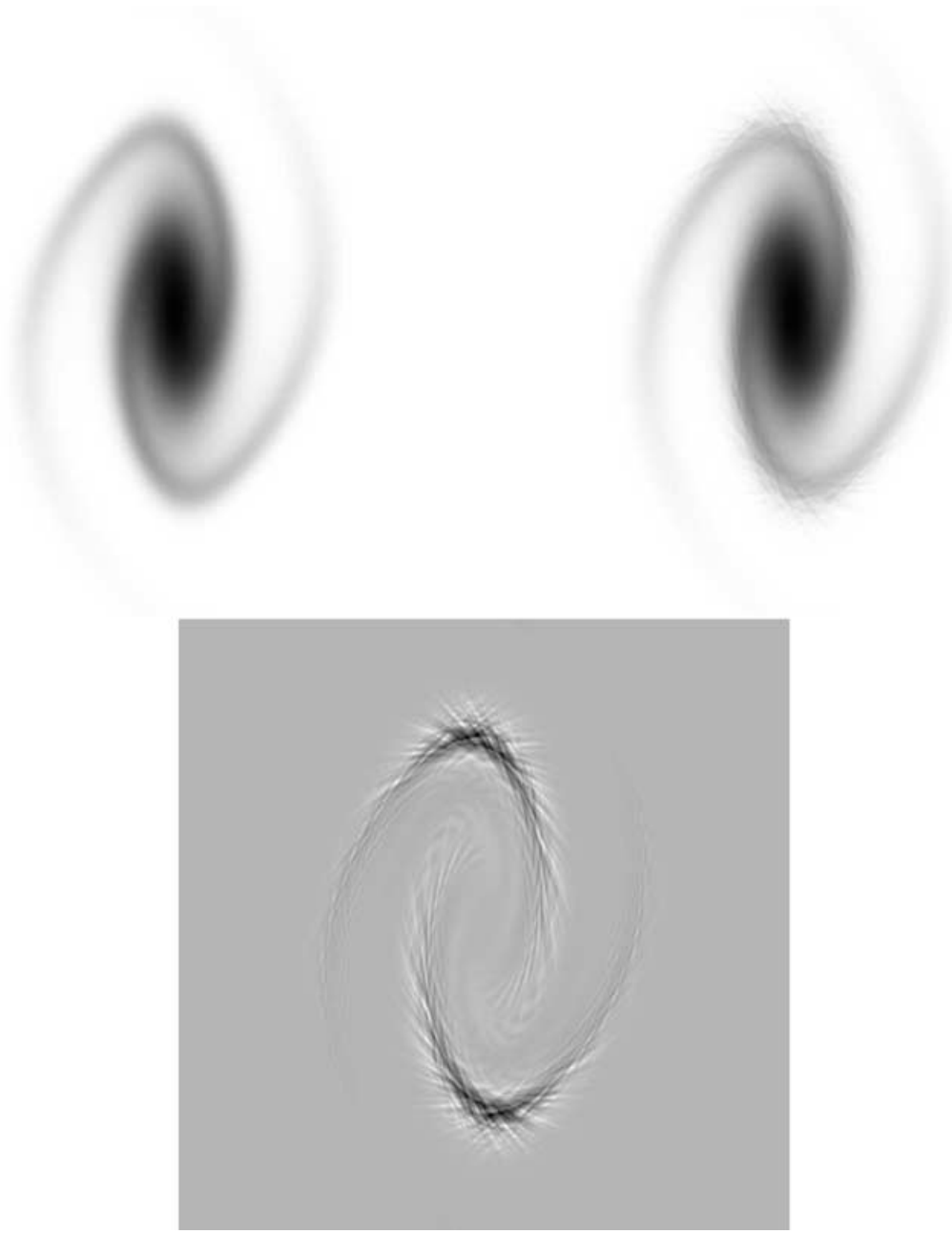

Figure 9. Apparition of a hairy structure after a few dynamical times. The simulation considered is the same as the one in the two left-hand panels of Fig. 8, at $t = 6$. The top left-hand panel gives the 'correct' answer, i.e. $f(x, v)$ obtained by remapping the density field with new round clouds using condition (85) with $E_{\max} = 0.05$. The top right-hand panel shows the 'wrong' answer obtained without remapping. The bottom panel displays the difference between left- and right-hand panels, which shows the hairy structure appearing due to the divergence of the axis ratios of the clouds.

trivial. The projected density of the clouds is simply calculated on a grid of interspacing Δ_s in such a way that the number of intersecting grid sites is always larger than N_s for any cloud. The force is then estimated exactly the same way as discussed in Section 3.2.2. In the current implementation, this fixed-resolution code is much faster than the tree-code, especially at the moment of remap, where a fast convolution method can be used to estimate the phase-space distribution function (Appendix A2). However, note that the tree-code part can still be considerably optimized for the remap part, since the fast convolution method can in principle be generalized to a non-structured grid.

3.3 Time-stepping implementation and diagnostics

In this section, we discuss our run-time implementation: a second-order predictor–corrector, as described in Section 3.3.1 This makes our approach second order both in time and in space. The determination of the slowly varying time-step is discussed in Section 3.3.2, as well as other diagnostics, such as energy conservation. Finally, Section 3.3.3 examines the critical issue of deciding when to perform a new sampling of the distribution function with a set of round clouds, by studying cumulative errors on the determination of the forces.

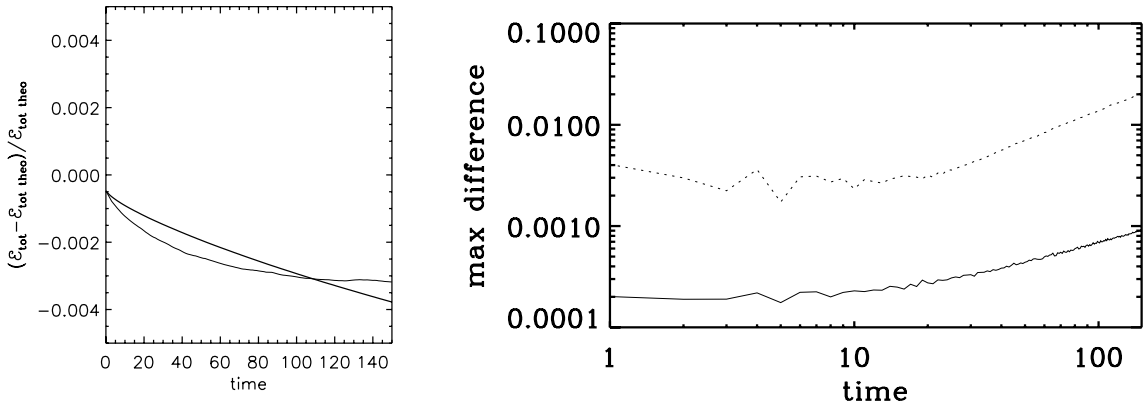

Figure 10. Left-hand panel: test of energy conservation for a stationary solution: The quantity $(\mathcal{E}_{\rm tot} - \mathcal{E}_{\rm tot\ theo})/\mathcal{E}_{\rm tot\ theo}$ is displayed as a function of time, where $\mathcal{E}_{\rm tot}$ is measured in our simulation according to equation (83) and $\mathcal{E}_{\rm tot\ theo}$ is given by equation (90). The parameters used to do the simulations are $\bar{\rho} = 2$ and $\sigma_v = 0.2$. The thick and the thin curves correspond to the high- and the low-resolution simulations, respectively. Right-hand panel: deviations from stationarity: the quantity $\max|\tilde{f}(x, v, t) - f_{\rm s}(x, v)|$ is represented as a function of time, where $\tilde{f}$ is the simulated f and $f_{\rm s}(x, v)$ is given by equation (87) with additional apodization as explained in the text. The continuous and the dotted curve correspond to the high- and the low-resolution simulations, respectively.

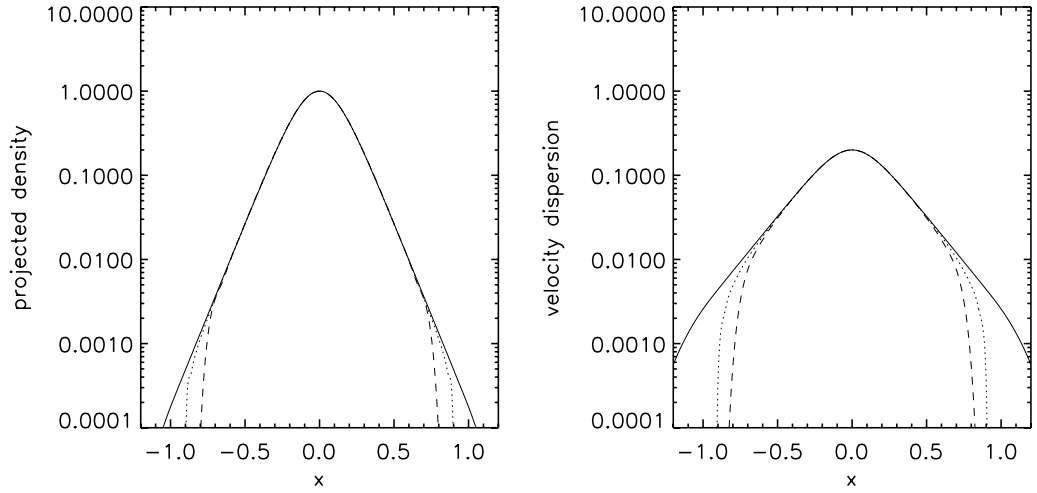

Figure 11. Mean projected density, $\rho(x) = \int f(x, v)\,{\rm d}v$ (left-hand panel) and velocity dispersion, $v_{\rm rms}(x)$, $v^2_{\rm rms}(x) \equiv \langle \delta v^2 \rangle = \int f(x, v) v^2\,{\rm d}v - [\int f(x, v) v\,{\rm d}v]^2$ (right-hand panel), as functions of position, for the stationary solution examined in Fig. 10. The solid curve corresponds to equation (87) with apodization as described in the text, which explains the bending of the tails visible in the right-hand panel. The dotted and the dashed curves give the results obtained from the high- and the low-resolution simulations, respectively.

3.3.1 Time-stepping implementation: global predictor–corrector

In our approach, we need a number of parameters to describe completely a set of clouds i, $i = 1, \ldots, N_{\rm tot}$, interacting through gravity. These parameters can be chosen as follows:

(i) the mass of the cloud, M_i: this one does not change with time, except when a remap with a new set of round clouds is performed;
(ii) the position of the centre of the cloud, $x^i_{\rm G}(t)$;
(iii) the velocity of the centre of the cloud, $v^i_{\rm G}(t)$;
(iv) the acceleration of the centre of the cloud, which we write $F_i(t)$;
(v) the parameter $\alpha^i_0(t)$ in equation (5), which is nothing but the projected density at position $[\, x^i_{\rm G}(t), v^i_{\rm G}(t)]$;
(vi) the function $w_i(t) = \sqrt{\Lambda^i_2(t)}$;
(vii) the time derivative of the function $w_i(t)$: $Dw_i(t) \equiv {\rm d}w_i/{\rm d}t$;
(viii) the area of the elliptical section of the cloud divided by π: $A_i = \bar{A}_i/\pi$; this does not change with time, except when a remap with a new set of round clouds is performed.

Recall that the parameters β^i_0, β^i_1 and β^i_2 defining the shape of the cloud are entirely determined by the function w_i, its time-derivative Dw_i, α^i_0 and $\bar{A}_i$, namely, from equations (22), (23) and (24), with the help of equation (20):

$$\beta^i_0 = (Dw_i)^2 + \frac{1}{(A_i w_i)^2}, \quad \beta^i_1 = -2w_i\ Dw_i, \quad \beta^i_2 = (w^i)^2. \quad (75)$$

To evolve the clouds we use a a standard second-order predictor–corrector algorithm, which is known to preserve simplectic behaviour quite well, a feature essential in phase space. To simplify the algorithm, we take the same global time-step for all the clouds. Our run-time implementation can be split up into the following six main parts.

(i) *Predictor step*:

$$x^i_{\rm G}(t_{n+1/2}) = x^i_{\rm G}(t_n) + \frac{1}{2}{\rm d}t_n v^i_{\rm G}(t_n), \quad (76)$$

$$w_i(t_{n+1/2}) = w_i(t_n) + \frac{1}{2}{\rm d}t_n Dw_i(t_n). \quad (77)$$

Figure 12. Phase-space distribution function at various times, for Gaussian initial conditions. Left- and right-hand panels correspond to the high- and the low-resolution simulations, respectively. Darker regions correspond to higher values of f.

(ii) *Force calculation*: the force is calculated as explained in Section 3.2.2 at the 'predicted' positions $x^i_G(t_{n+1/2})$ within each cloud of 'predicted' projected density given by equation (70) using $\Lambda_2(t_{n+1/2}) = [w_i(t_{n+1/2})]^2$. This gives, for each cloud, the parameters $F_i(t_{n+1/2})$ and $\alpha^i_0(t_{n+1/2})$.

(iii) *Corrector step*:

$$x^i_{\rm G}(t_{n+1}) = x^i_{\rm G}(t_{n+1/2}) + \frac{1}{2}\,{\rm d}t_n v^i_{\rm G}(t_n) + \frac{1}{2}\,{\rm d}t_n^2 F_i(t_{n+1/2}), \tag{78}$$

$$v^i_{\rm G}(t_{n+1}) = v^i_{\rm G}(t_n) + {\rm d}t_n F_i(t_{n+1/2}), \tag{79}$$

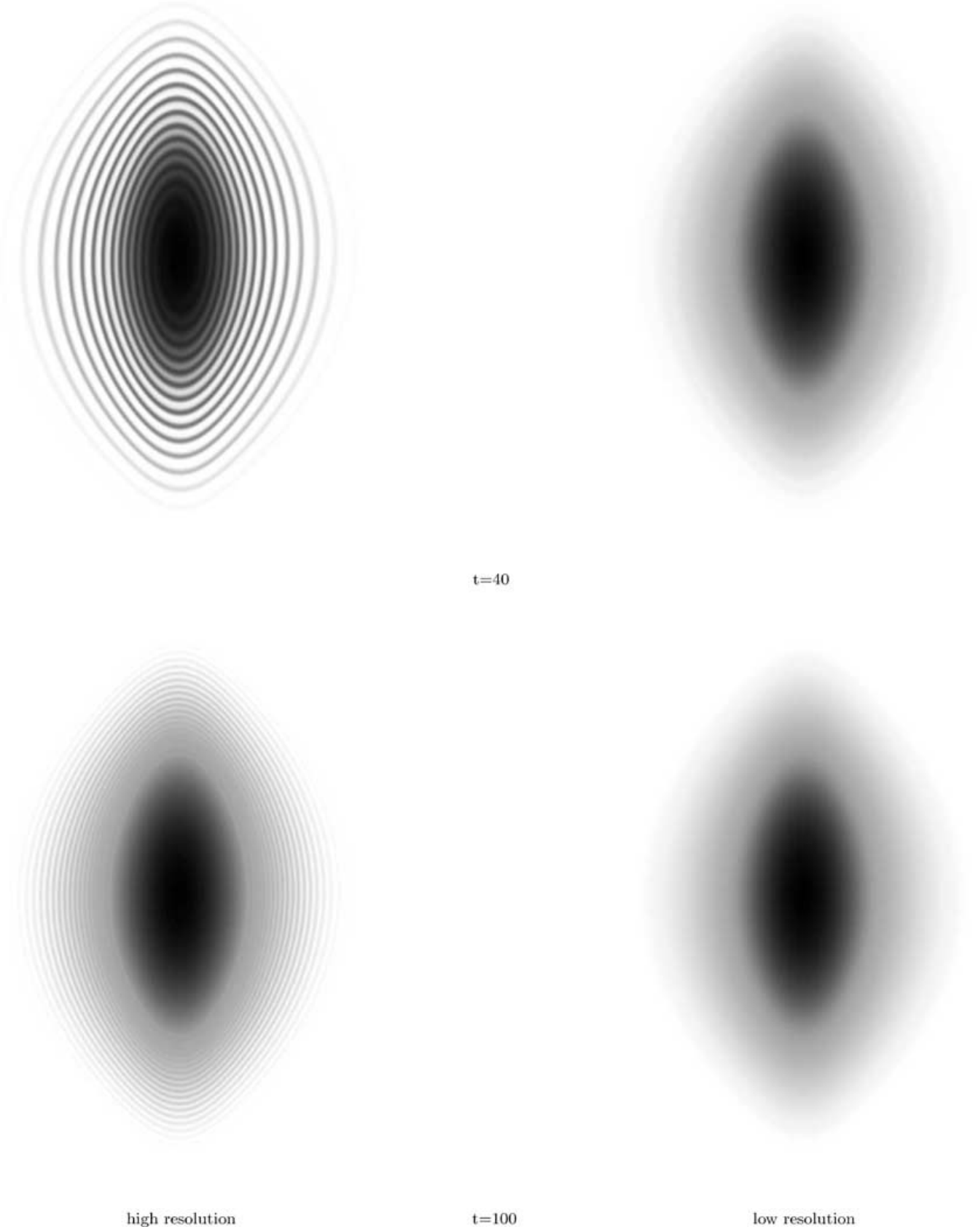

Figure 12 – *continued*

$$w_i(t_{n+1}) = w_i(t_{n+1/2}) + \frac{1}{2}\,\mathrm{d}t_n Dw_i(t_n) + \frac{1}{2}\mathrm{d}t_n^2\left[\frac{1}{A_i^2 w_i^3(t_{n+1/2})} - 2\alpha_0(t_{n+1/2})w_i(t_{n+1/2})\right], \quad (80)$$

$$Dw_i(t_{n+1}) = Dw_i(t_n) + \mathrm{d}t_n\left[\frac{1}{A_i^2 w_i^3(t_{n+1/2})} - 2\alpha_0(t_{n+1/2})w_i(t_{n+1/2})\right]. \quad (81)$$

(iv) *Outputs*: various quantities such as all the information on the clouds, the density in phase space, etc. can be output at this point.

(v) *Diagnostics: calculation of next time-step, test for Lagrange remap, energy conservation check:* as we are going to describe in more detail below, we use a time-step constrained by the slope of the force, $-2\alpha_0^i(t_{n+1/2})$. The fact that the next time-step, $\mathrm{d}t_{n+1}$, depends on a quantity calculated half a time-step before does not affect significantly the quasi-simplectic nature of our integrator. The important thing here is that $\mathrm{d}t_n$ should vary slowly with time. At this point, we also test if it is necessary to remap the distribution function with a new set of round clouds, due to the accumulation of deviations from local harmonicity of the potential, as studied in Section 3.3.3.

(vi) *Lagrange remap, if needed*, as explained in Section 3.1.

Note importantly, thus, that our algorithm is second order both in space and in time. It is quasi-simplectic in the sense that it would be time reversible, equivalent to generalized leap-frog, if the time-step was constant.

3.3.2 Diagnostics: 'Courant' condition and energy conservation

There is a list of diagnostics to perform during run-time. The most important are those related to time-stepping. What matters in our Lagrangian approach is that orbits in phase space should be sampled with sufficient number of points as well as cloud-shape variations during the trajectory. From the analysis of Section 2.2.1, this means, using equation (37),

$$\mathrm{d}t \leqslant \frac{\pi\mathcal{C}}{\max_i \sqrt{2\alpha_0^i}}, \quad (82)$$

where $\mathcal{C}$ is a 'Courant' parameter small compared to unity. In practice we find that we should have $\mathcal{C} \lesssim 0.01$. For all the simulations made in this paper, we took $\mathcal{C} = 0.01$.

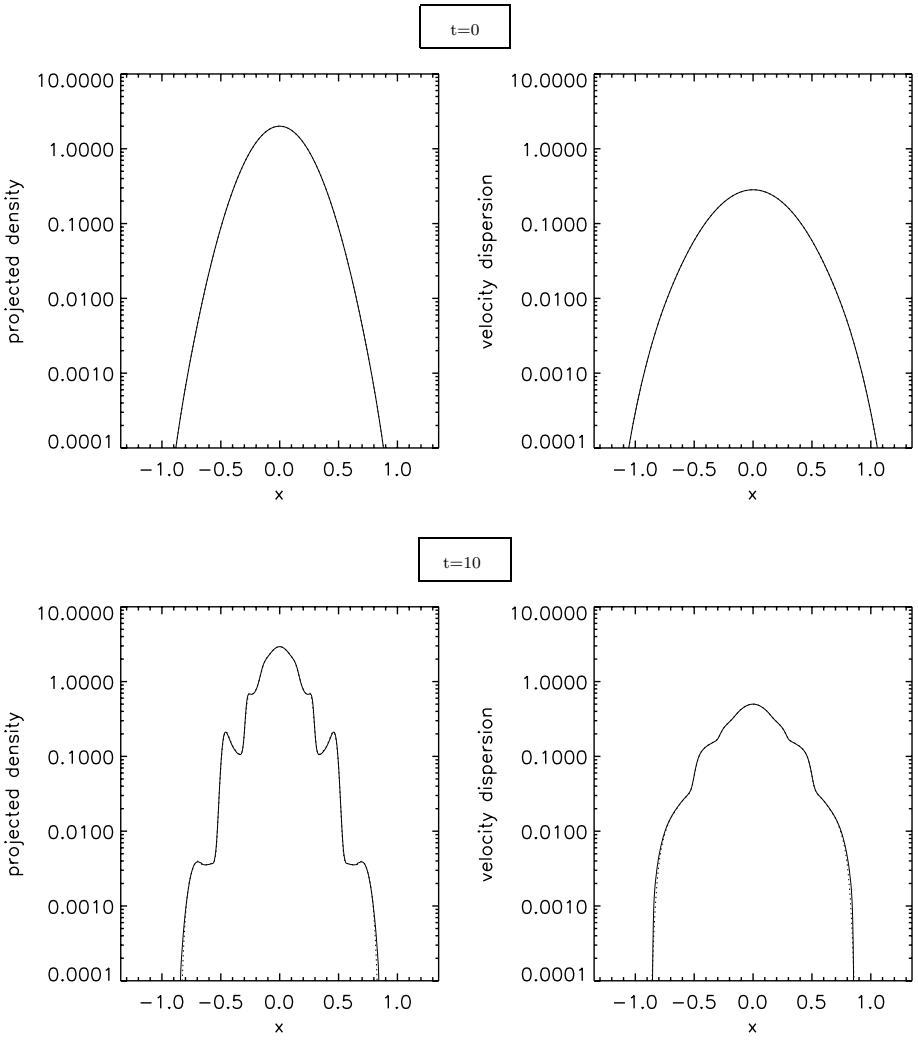

Figure 13. Mean projected density (left-hand panels) and velocity dispersion (right-hand panels), as functions of position, for the simulations with Gaussian initial conditions. Each row of panels corresponds to a given time. The solid curve and the dotted curve are for the high- and the low-resolution simulations, respectively, except for $t = 0$: in that case, the curves are calculated according to equation (84).

Another diagnostic is of course total energy conservation, which is, in our units,

$$\mathcal{E}_{\rm tot} \equiv \frac{1}{2}\int v^2 f(x,v,t)\,{\rm d}x\,{\rm d}v - \frac{1}{4}\int\left[\left(\frac{{\rm d}\phi}{{\rm d}x}\right)^2 - M^2\right]{\rm d}x = \text{constant}. \quad (83)$$

This equation assumes that the system is finite of total mass M. We added a term proportional to M^2 to compensate for the divergence in the integral $\int({\rm d}\phi/{\rm d}x)^2\,{\rm d}x$ so that the constant in equation (83) is finite.

One might use instantaneous deviations from energy conservation to constrain the time-step, but we chose here to test energy conservation only as a consistency check. We shall indeed see in Section 4 that energy is very well conserved in our code.

Finally notice that mass is conserved by definition, except when a resampling of the distribution is performed with a new set of round clouds. During this step, mass conservation is enforced as explained in Section 3.1.4, as well as total momentum.

3.3.3 Lagrange remap

The assumption of a quadratic local potential is in general always violated to some extent during run-time since it corresponds to variations in the projected density $\rho(x)$ (Section 2.2.4). What counts, as already mentioned in Section 3.2.2, is the cumulative error on the force, which translates into clouds with increasingly wrong ellipticity parameters. This effect, even if always small during a time-step (see Section 2.2.4), cumulates with time. In particular, the clouds get inevitably spuriously elongated and some 'hairy' structure appears during evolution (Fig. 9). This is why we have at some point to remap the distribution function with a new set of round clouds, an operation that we call 'Lagrange remap'.

To illustrate this point, we consider here the same example as in Section 2.2.4: the evolution of a distribution function initially Gaussian, i.e. given by

$$f(x,v) = \bar{\rho}\,\exp\left(-\frac{1}{2}\frac{x^2+v^2}{h^2}\right), \quad x^2+v^2 \leqslant \mathcal{R}^2,$$
$$f(x,v) = \frac{1}{2}\bar{\rho}\,\exp\left(-\frac{1}{2}\frac{x^2+v^2}{h^2}\right) \times\left\{\cos\left[\frac{\pi}{2}(\sqrt{x^2+v^2}-\mathcal{R})/\mathcal{R}_{\rm apo}\right]+1\right\}, \quad (84)$$
$$x^2+v^2 \leqslant (\mathcal{R}+2\mathcal{R}_{\rm apo})^2.$$

This initial profile will be used for the set of simulations studied in Section 4. The same apodization as in equation (65) is performed to regularize the function at its edges: equation (84) approaches

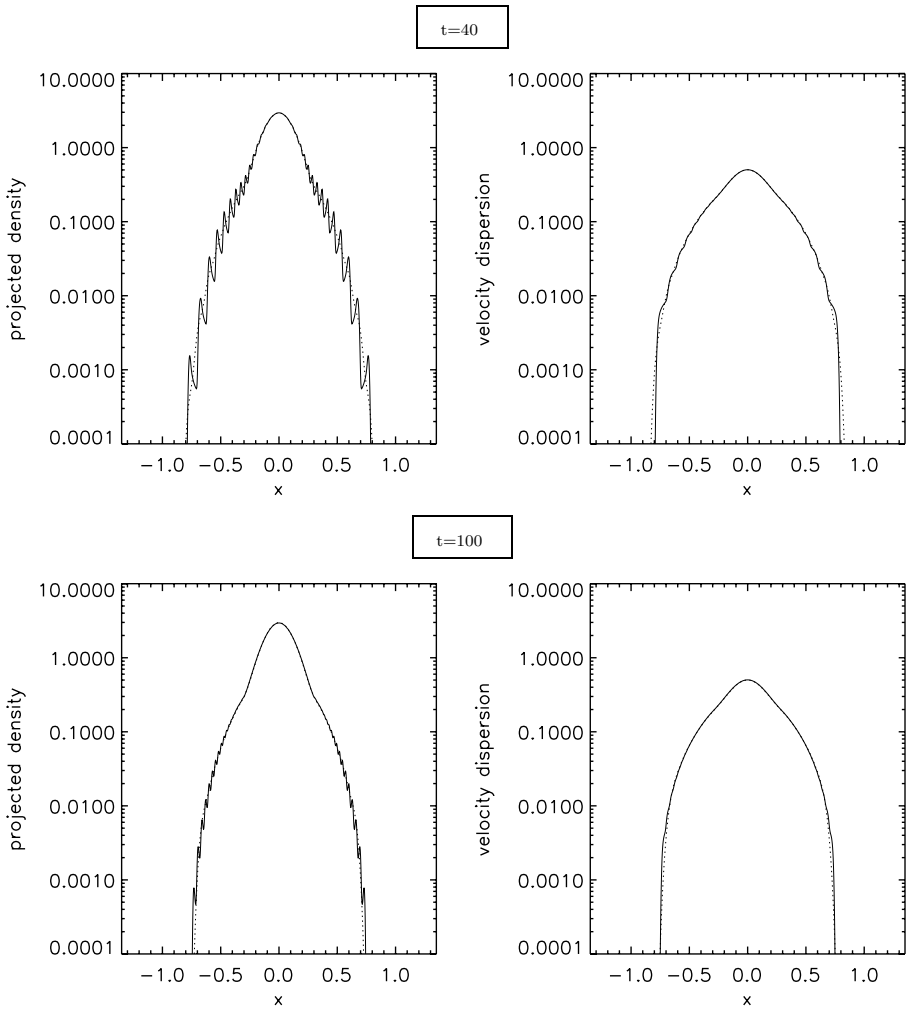

Figure 13 – *continued*

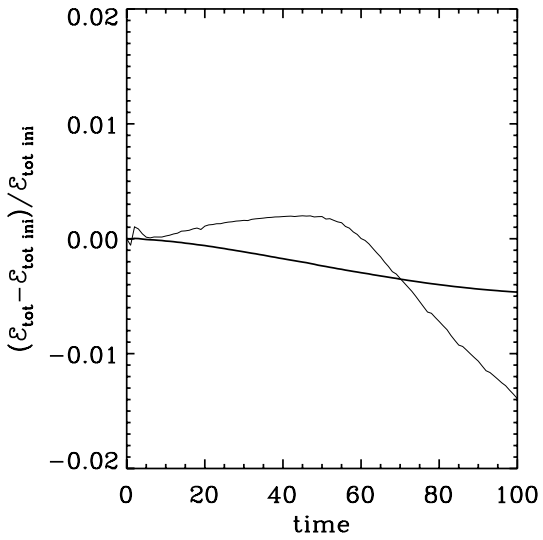

Figure 14. Energy conservation for the simulations with Gaussian initial conditions. The relative deviation from energy conservation is displayed as a function of time for the high-resolution simulation (thick curve) and the low-resolution simulation (thin curve): $\mathcal{E}_{\rm tot}$ and $\mathcal{E}_{\rm tot\ ini}$ correspond to total energy and initial total energy, respectively, as measured in the simulations according to equation (83).

Gaussian initial conditions only if $h \ll \mathcal{R}$. For the simulation considered here, we take $h = 0.2$, $\bar{\rho} = 1/(2\pi h^2)$, $\mathcal{R} = 1$, $\mathcal{R}_{\rm apo} = 0.1$. The cloud size is $R = h/10 = 0.02$, similarly as in Section 2.2.4 and the intercloud spacing is thus given by $\Delta_{\rm g} = \sqrt{2}R = 0.028$. To compute the force and the error on it (equation 73), we take $N_{\rm s} = 8$ as advocated in Section 3.2.2.

The left-hand panel of Fig. 8 gives the maximum instantaneous error on the force as a function of initial position, $x_{\rm ini}$. It shows that the deviations from the local quadratic approximation globally increase with time, as expected. What is important here, though, is the accumulation, $E_{\rm cum}$, of little kicks at each time-step due to these deviations, as given by equation (74). This quantity, shown as a function of time in the middle panel of Fig. 8, has the dimensionality of a velocity. It should remain small compared to the velocity resolution:

$$\frac{E_{\rm cum}^{\rm max}}{\Delta_v} \leqslant E_{\rm max} \ll 1. \tag{85}$$

If condition (85) is violated, a new sampling of the distribution function with round clouds should be performed. We notice that a typical value of the threshold, $E_{\rm max} = 0.05$, i.e. a 5 per cent cumulative error, corresponds, as expected, to a fraction of dynamical time, $t \simeq 0.4$. Interestingly, but not surprisingly, if condition (85) is enforced, the number $n_{\rm remap}$ of time-steps between each remap is pretty stable, as shown in the right-hand panel of Fig. 8. For our Courant parameter choice $\mathcal{C} = 0.01$ and for $E_{\rm max} = 0.05$ we find $n_{\rm remap} \simeq 15$. Its variations decrease with time, as expected, while the system converges to a stationary state. As a result, it is a very good approximation to keep $n_{\rm remap}$ fixed, which is quite useful, as it is not necessary in that case to estimate the cumulative error, and as a consequence, $N_{\rm s} = 2$ can be taken to speed up considerably the calculation of the force, without any significant loss in accuracy.

One would thus be tempted to fix $n_{\rm remap}$ as a function of our Courant condition using the following formula,

$$n_{\rm remap}(\mathcal{C}) = \frac{n_{0.01}}{(\mathcal{C}/0.01)}, \tag{86}$$

with, typically, $n_{0.01} \simeq 15$ to produce a cumulative error on the force of the order of 5 per cent. This estimate is only valid for the initial conditions considered here, but according to the arguments of Section 2.2.4, the same result should be roughly found for other configurations: with $\mathcal{C} \simeq 0.01$, a Lagrange remap should be performed every 10–15 time-steps.

One might be worried that a cumulative error of a few per cent per Lagrange remap is too much: accumulation of errors after a few thousand remaps might still be too large. However, on has to take into account the fact that the cumulative error, as given by equation (74), does not take into account possible global cancellations during the dynamics. In particular, if a given symmetric potential is fixed, we see that a cloud plunged in the right side of the potential will be the object of next to quadratic distortions opposite to those in the other side: they will cancel each other, which makes the solution rather stable on the long term, even if our absolute cumulative error becomes very large. This is illustrated in Fig. 9, which compares the results obtained for the simulation of Fig. 8 without remapping at

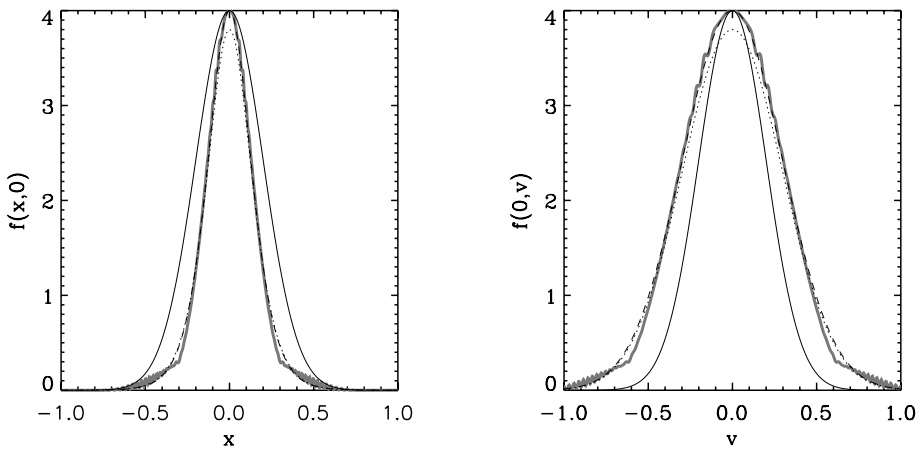

Figure 15. Comparison of the simulation with Gaussian initial conditions with a stationary solution. The function $f(x, 0)$ is displayed in the left-hand panel, while the function $f(0, v)$ is displayed in the right-hand panel. On each panel, the simulation (thick grey curve) is compared to the stationary solution given by equation (87) with the same energy and same mass (dots). The dashes are the same as the dots, except that the stationary solution has been normalized to match the maximum of f. For reference, the thin solid curve corresponds to initial conditions.

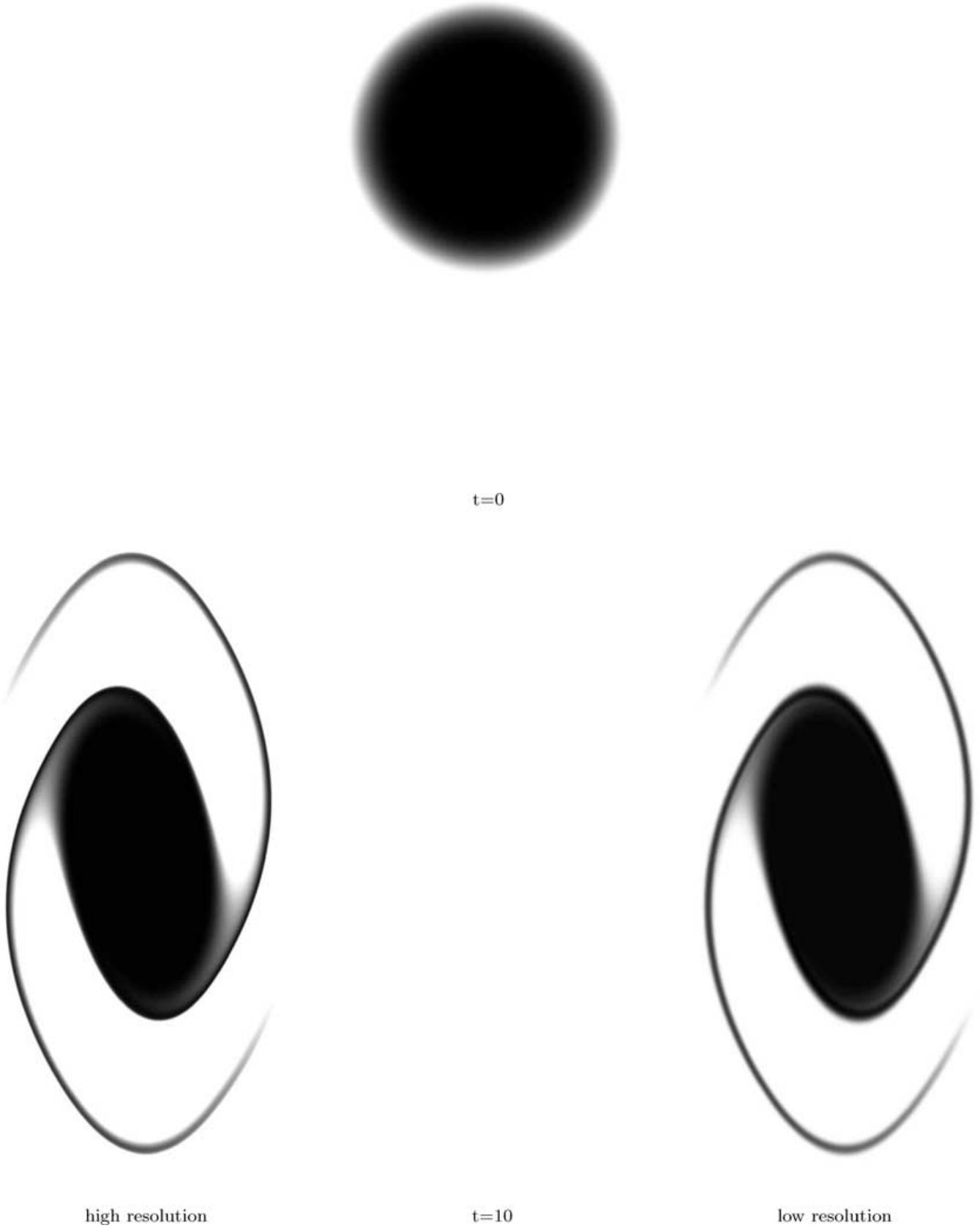

Figure 16. The phase-space distribution function for top-hat initial conditions, similarly as in Fig. 12.

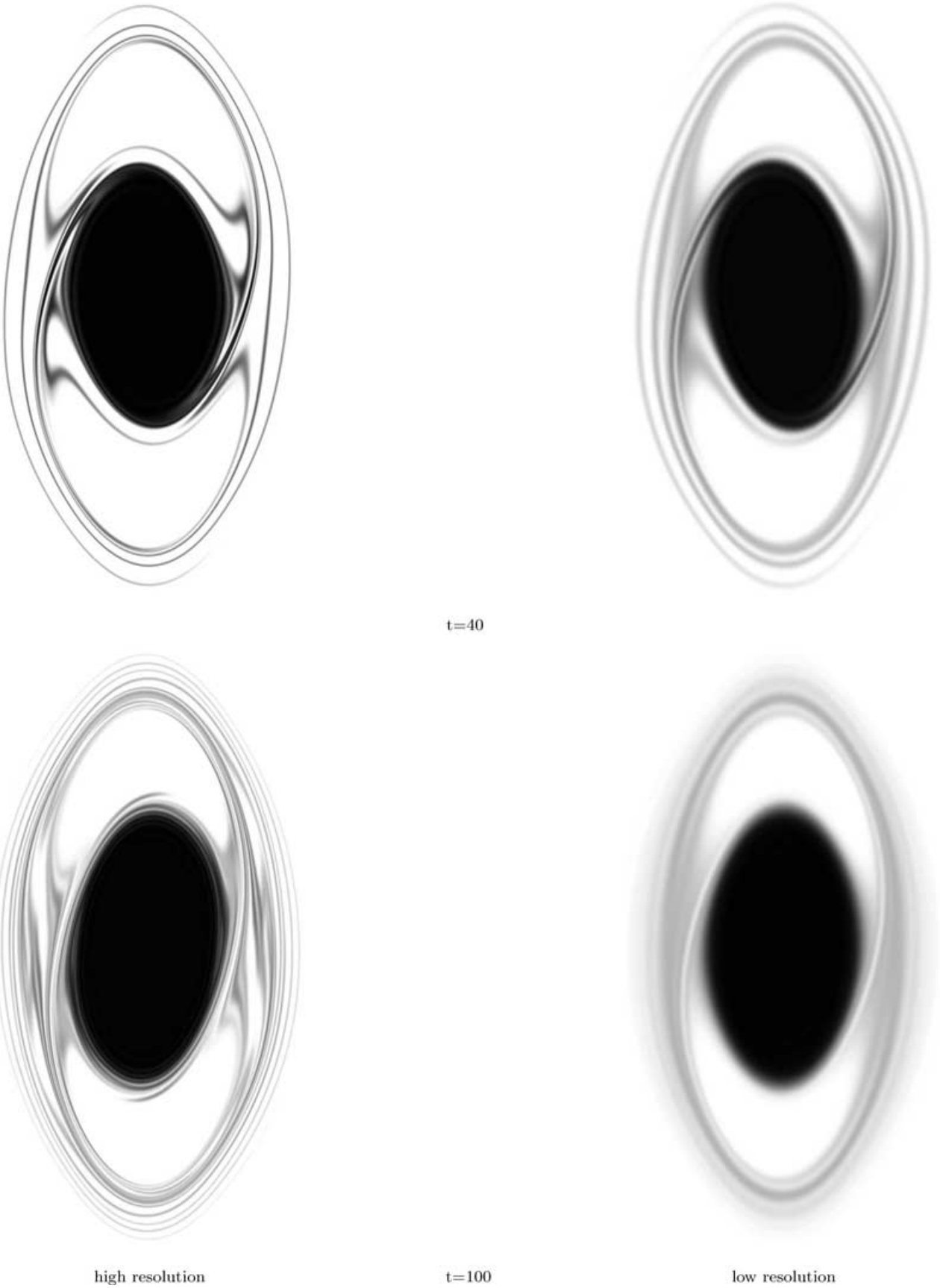

Figure 16 – *continued*

$t = 6$ to the exact solution. The 'hairy' effect is quite visible, but the solution is rather close to the correct answer at the coarse level.

4 SOME APPLICATIONS

In this section, we show how our tree-code performs for three different kinds of initial conditions. Similar results should be obtained for the CM code, so we do not feel it necessary to show them here.

The first model, considered in Section 4.1, is a stationary profile, satisfying $\partial f/\partial t = 0$. This is a crucial test: the code should be able to maintain a stationary profile during many dynamical times to a very good accuracy. We shall see that our code passes this test with great success.

The second model, considered in Section 4.2, is a Gaussian distribution function: this kind of very smooth initial conditions is well adapted to our method, which should be able to follow their evolution quite accurately until the details appearing during run-time become too small to be resolved by our sampling clouds. We shall see that at this point our numerical solution still gives a very good coarse-level version of the full-resolution one.

The third model, considered in Section 4.3, is a top-hat apodized by a cosine. The evolution of this a kind of distribution function allows us to test directly to what extent our code can maintain a region with $f =$ constant. In particular, effects of aliasing can be quantified easily.

For each of the models considered above, we perform one 'low resolution' simulation with $\Delta_x = \Delta_v = 0.02$, and one 'high resolution' simulation with $\Delta_x = \Delta_v = 0.005$. We run these simulations using a Courant condition $\mathcal{C} = 0.01$ and a Lagrange remap every 15 time-steps, following the conclusions of the analyses above. To perform the resampling, we use the Lucy algorithm with 10 iterations exactly, whatever the residue obtained between the resampled distribution and the previous one, following discussions of Section 3.1.2. To maintain the domain of contributing values of f finite, we proceed as explained in the paragraph just after equation (63) with $\delta_{\min} = 5 \times 10^{-5}$. In all cases, we aim to evolve the system during approximately 30 dynamical times (corresponding to 60 orbital times), a goal that we achieve in the following way. We set up initial conditions such that ρ_0 is of order two in equation (37) for the top-hat and the Gaussian case, and evolve them up to $t = 100$. For the stationary solution, the choice of ρ_0 is of order unity and we evolve the simulation up to $t = 150$. With our Courant condition, this amounts finally to approximately 7000 time-steps for the stationary simulation and 8000 time-steps for the simulations with Gaussian and top-hat initial conditions, corresponding to a large number of remaps, about 470 and 530, respectively.

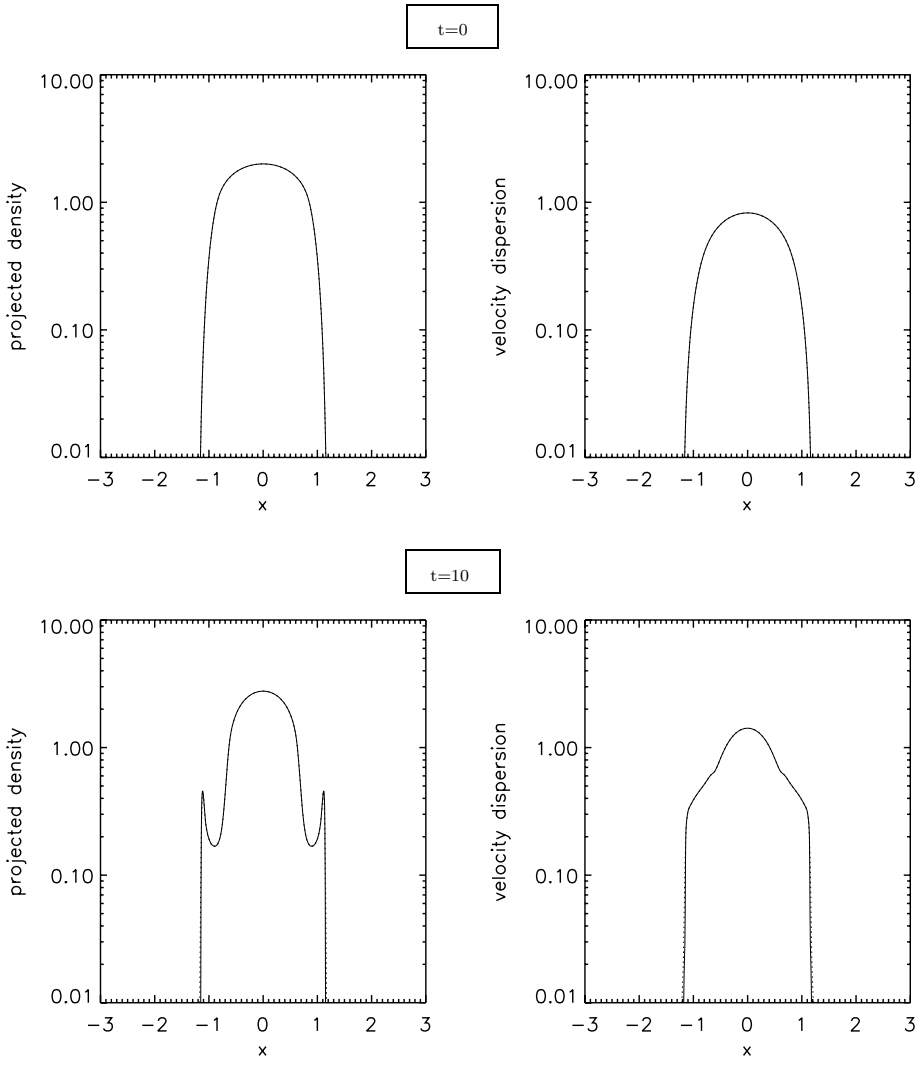

Figure 17. Mean projected density and velocity dispersion, for the simulations with top-hat initial conditions, similarly as in Fig. 13.

This way we are able to quantify the limitations of our approach, namely coarse-graining and aliasing effects due to the finite size of the sampling clouds at the moment of remap. We shall see that, as expected, the low-resolution simulations give a very accurate coarse-grained version of the high-resolution ones.

4.1 First application: stationary profile

A family of stationary solutions is given by (Spitzer 1942)

$$f_{\rm s}(x,v)=\bar{\rho}\left[\cosh\left(\frac{x}{\sigma_x}\right)\right]^{-2}\exp\left(-\frac{1}{2}\frac{v^2}{\sigma_v^2}\right), \tag{87}$$

with

$$\sigma_x=\sqrt{\frac{\sigma_v}{\sqrt{2\pi}\bar{\rho}}}. \tag{88}$$

For further reference, the total mass is

$$M=2(2\pi)^{1/4}\bar{\rho}^{1/2}\sigma_v^{3/2}, \tag{89}$$

and the total energy (equation 83) is

$$\mathcal{E}_{\rm tot}=3(2\pi)^{1/4}\bar{\rho}^{1/2}\sigma_v^{7/2}. \tag{90}$$

To perform the simulations, we set up initial conditions with the same apodization as in equations (65) and (84), to make the support of the function compact. To achieve an initial set-up sufficiently close to the stationary solution, we must have σ_v and σ_x sufficiently small compared to the apodization radius, $\mathcal{R}$. Our choice here is $\mathcal{R}=1, \mathcal{R}_{\rm apo}=0.2, \bar{\rho}=2, \sigma_v=0.2$ [hence, $\sigma_x\simeq 0.2\simeq\sigma_v$]. With this set-up, the number of clouds contributing during runtime is about 80 000–120 000 and 4200–7500 in the high- and low-resolution simulations, respectively. This number decreases slowly with time as a result of the truncation procedure used to maintain the support of f compact (see Section 3.1.2). In principle we could fine tune this procedure in order to maintain the number of sampling clouds approximately constant with time, but we decided not to do so at this point, because it affects only the tails of the distribution function.

Fig. 10 examines energy conservation and deviations from the stationary solution as functions of time. As expected, energy conservation is very good both for the high- and for the low-resolution simulations, better than 0.5 per cent. However, it worsens with time. This is mainly a consequence of our truncation of the phase-space distribution function at $\delta_{\rm min}=5\times10^{-5}$. Note that the maximum difference between the true distribution function and the simulated one tends to augment linearly with time at a

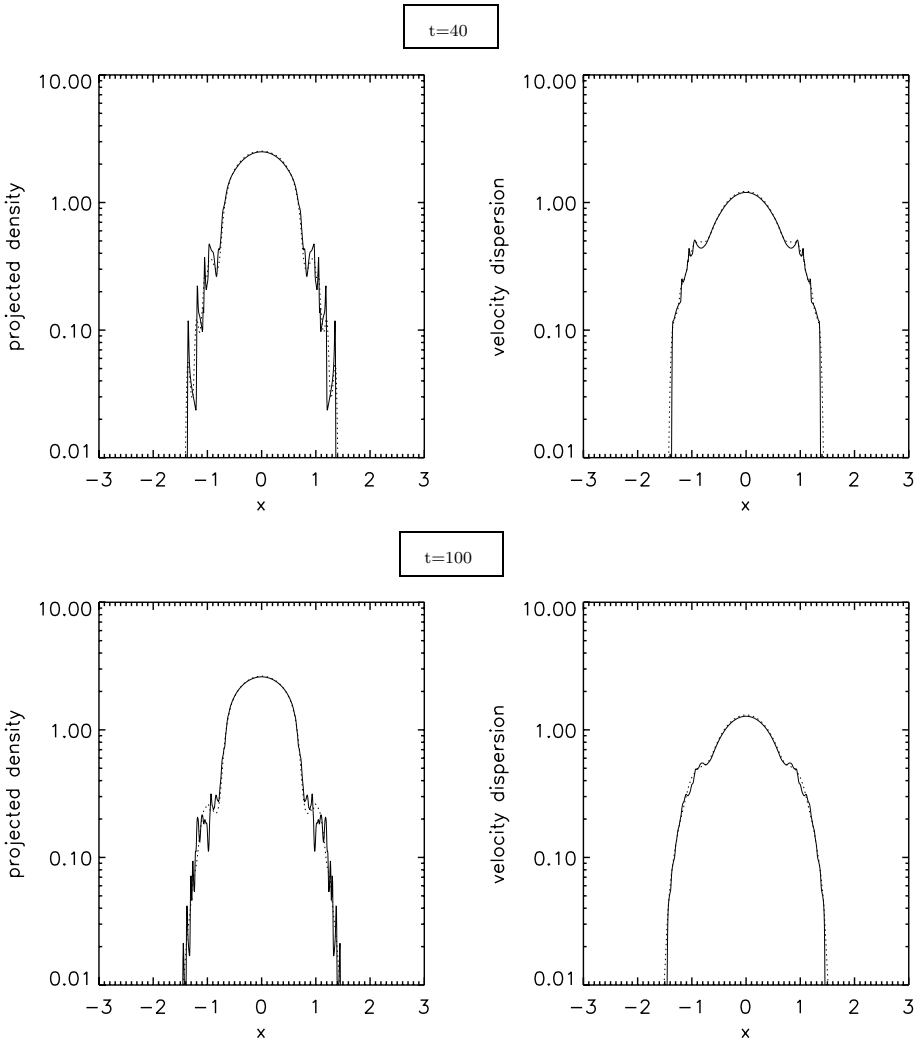

Figure 17 – *continued*

certain point, even for the high-resolution simulation, as illustrated by the right-hand panel of Fig. 10. For the high-resolution simulation, this is again mainly due to the truncation of the tails of f. Indeed, the difference δf between the numerical and the analytical solutions measured at the centre of the fluctuation, $(x, v) = (0, 0)$, is only of the order of -2×10^{-4} for $t = 150$. For the low-resolution simulation, another effect adds up to the truncation of the tails, namely a weak diffusion at the maximum of f at the moment of Lagrange remap. When fluctuations become of the order of the cloud size, one can indeed expect two competing effects, according to the details of the shape of the sampled distribution function: (i) a coarse-graining effect, where small-scale fluctuations tend to be progressively smeared out, and (ii) an aliasing effect already discussed in Section 3.1.2, where, on the contrary, artificial contrasts are created with the appearance of spurious oscillations. Since the distribution function we sample here is smooth, effect (ii) is not present. However, although a deconvolution method such as Lucy's (or van Citter's) aims to minimize effect (i), it cannot reduce it completely. This effect is very small, since it induces a rather weak diffusion, even for the low-resolution simulation, of the order of 2 per cent only at the centre of the fluctuation at $t = 150$.

Fig. 11 shows the zeroth and second moments of $f(x, v)$ with respect to velocity, namely the projected density and the velocity dispersion, as functions of x, at latest output, $t = 150$. Even though the low-resolution simulation has already significantly diffused, with a two per cent depletion on the maximum of f, the overall solution is still pretty close to the correct one. The deviations observed in the tails both in the high- and the low-resolution simulations are a consequence of the truncation procedure used to maintain the computing domain finite.

4.2 Second application: a Gaussian as initial conditions

The Gaussian initial conditions we consider are specified by equation (84), with $\mathcal{R} = 0.8$, $\mathcal{R}_{\rm apo} = 0.2$, $\bar{\rho} = 4$ and $h = 0.2$. With this set-up, the number of clouds contributing is roughly 95 000–120 000 and 6600–7700 for the high- and the low-resolution simulations, respectively.

Fig. 12 shows the distribution function in phase space at various times, $t = 0$, 10, 40 and 100. As expected, a spiral structure appears, which roles up with time. At some point, it becomes so thin that it disappears due to finite resolution. Of course, this event happens earlier for the low-resolution simulation. Visually, this latter seems to represent a very good coarse-grained version of the high-resolution simulation. This can be examined in more detail in Fig. 13, which displays projected density and velocity dispersion at the same instants as in Fig. 12. At $t = 10$, it is not possible to distinguish yet between low and high resolution. The difference between the two simulations is the most significant for $t = 40$: in that case, details are lost in the low-resolution simulation, but it gives a rather accurate coarse-grained version of the high-resolution simulation. At late time, $t = 100$, details have nearly disappeared even in the high-resolution simulation and the agreement between low and high resolution remains excellent.

Fig. 14 displays the relative deviation from energy conservation as a function of time. For the high-resolution simulation, results are very similar to what was obtained for the stationary solution: total energy decreases slowly with time, mainly because of our truncation of the tails of the phase-space distribution function, but is conserved with a precision better than 0.5 per cent at $t = 100$. The behaviour observed for the low-resolution simulation is more complex. Indeed, energy first increases up to $t \simeq 50$, due to coarse-graining effects which introduce a slight bias in the tails of the distribution function. At variance with the high-resolution simulation, these effects dominate over those due to the truncation until details in the distribution have completely disappeared. Note that the maximum of f, which should stay constant, presents small variations between initial and final time. The net result is an increase of 0.1 and 2 per cent for the high- and the low-resolution simulations, respectively. This is probably due to aliasing effects, which are, however, not expected to affect energy conservation significantly.

Examination of the second part of Fig. 13 suggests that the system is relaxing towards a stationary regime. Fig. 15 compares the phase-space distribution function at the latest stage of the high-resolution simulation to a stationary solution given by equation (87). To compute the parameters of the stationary solution, we use equations (89) and (90). For the total energy, we take the initial value obtained from the (apodized) Gaussian profile and obtain $\bar{\rho} \simeq 3.79$ and $\sigma_v = 0.298$, which corresponds to the dotted curve. It globally agrees well with the simulated solution (thick grey curve), if ones keeps initial conditions as a reference for comparison (solid thin curve). Note that the agreement is improved in the high f part if $\bar{\rho}$ is changed to 4 (dashed curve), but remains imperfect: the dashed curve is not able to reproduce the clear transition between a plateau at low values of f and the central belt shape, which suggests a more complex stationary regime, with two components.

4.3 Third application: a top-hat as initial conditions

To set up initial conditions, we use equation (65) with $\mathcal{R} = 0.8$, $\mathcal{R}_{\rm apo} = 0.2$ and $\bar{\rho} = 1$. The number of clouds contributing to f varies between 190 000 and 350 000 and between 11 000 and 29 000 for the high- and low-resolution simulations, respectively.

Figs 16 and 17 show the resulting distribution function in phase space and its zeroth and second moments, at various times, $t = 0$, 10, 40 and 100, both for the low- and the high-resolution simulations. The system builds with time a two component structure: most of the region $f = \bar{\rho}$ remains in a compact structure with a roughly elliptic shape rotating and pulsating around the centre, while a ring appears around it by the effect of rolling up. The appearance of this ring,

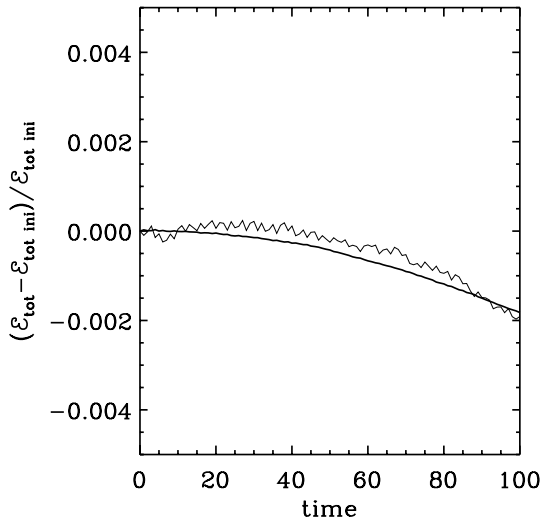

Figure 18. Energy conservation for the simulations with top-hat initial conditions, similarly as in Fig. 14.

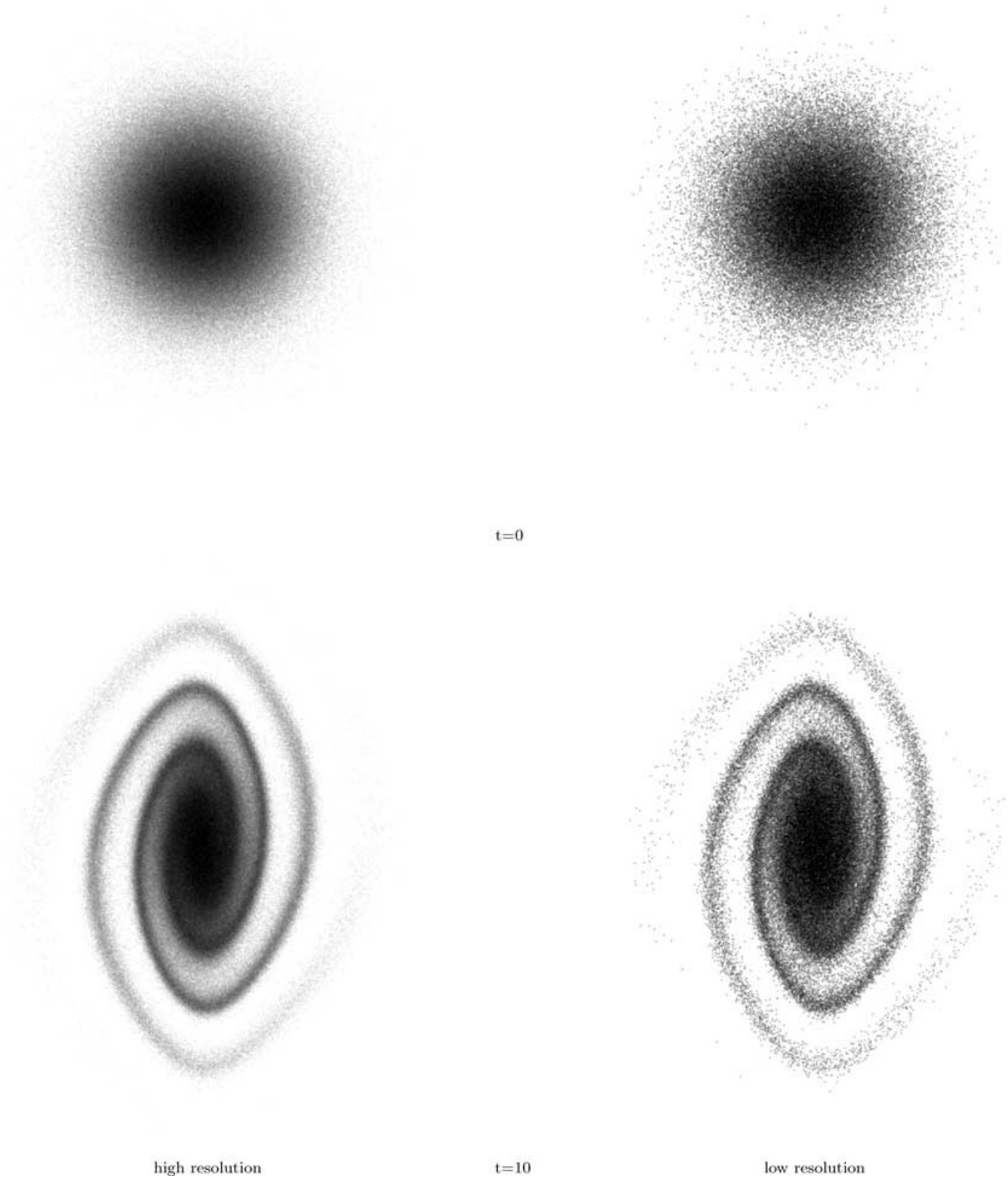

Figure 19. Phase-space distribution function at various times, for Gaussian initial conditions in the N-body simulations. Left- and right-hand panels correspond to the high- and the low-resolution simulations, respectively. Darker regions correspond to higher values of f. This figure can be compared directly with Fig. 12, which was obtained with the cloud method.

formed of a very fine spiral structure transporting a small fraction of the total mass (see Fig. 17), explains the significant increase of the number of clouds contributing to the distribution function. At $t = 100$, our pulsating core+ring structure has not converged to any stationary solution, as expected, but remains topologically stable on the coarse level, if one takes into account the fact that mass migrates slowly from the edges of the central patch to the external ring.

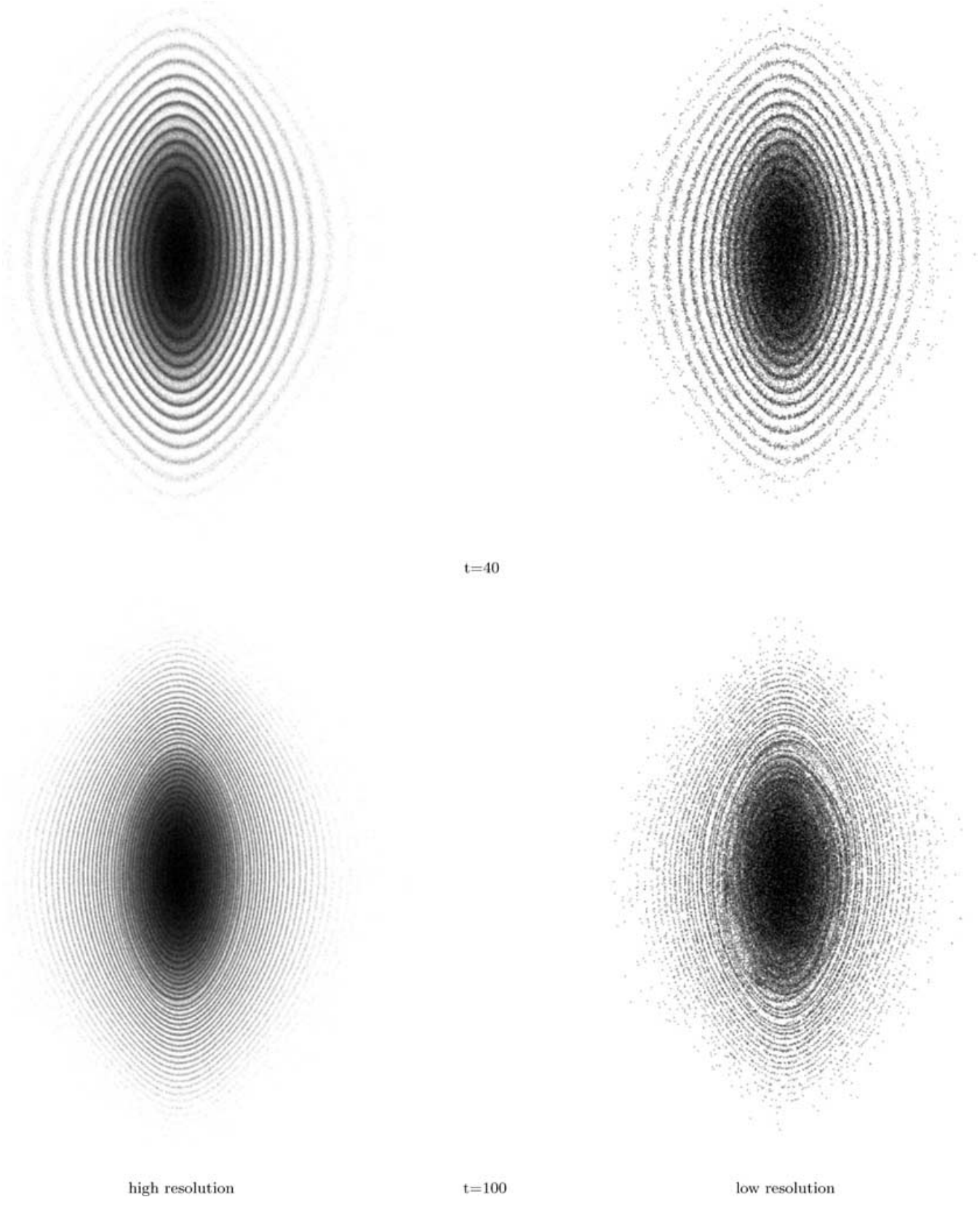

Figure 19 – *continued*

A careful examination of Figs 16 and 17 shows that the agreement between low and high resolution is pretty good, although not as impressive as for the Gaussian case: some details differ even at the coarse level, particularly in the region of transition between the external ring and the central patch. These differences do not affect energy conservation significantly, as illustrated in Fig. 18: in both simulations, energy decreases slowly with time, mainly as a result of our truncation of the tails of the phase-space distribution function, and is conserved with an accuracy better than 0.2 per cent.

Note finally that aliasing effects seem to be more significant here than for the Gaussian simulation, although still quite reasonable: at $t = 100$, we find for both simulations that the maximum of f is about 1–2 per cent larger than it should be. This happens at the edges of the central patch. At the centre, we measure $f(0, 0) \simeq 1.005$ and 1.0007 for the low- and the high-resolution simulations, respectively, which represents a very small deviation from unity.

5 COMPARISON WITH STANDARD *N*-BODY

In this section we show that the cloud method matches very well the results obtained with with the standard N-body approach, but with a significant gain in accuracy and smoothness, as expected. To conduct our analyses, we concentrate on the high-resolution simulation with Gaussian initial conditions studied in Section 4.2.

We wrote a simple particle-mesh code (PM) similar to the CM code described in Section 3.2.3 In our N-body code, the projected density is thus calculated by cloud-in-cell interpolation (see, e.g. Hockney & Eastwood 1988) of the mass of each particle on a fine mesh of step Δ_g. Our choice of mesh interspacing is $\Delta_g = 200/N$, where N is the number of particles. The initial conditions are exactly the same as in Section 4.2, but no apodization was performed (equation 84). Instead, we just put no particle beyond $\mathcal{R}$. This does not make much difference, since the cut-off radius $\mathcal{R}$ corresponds

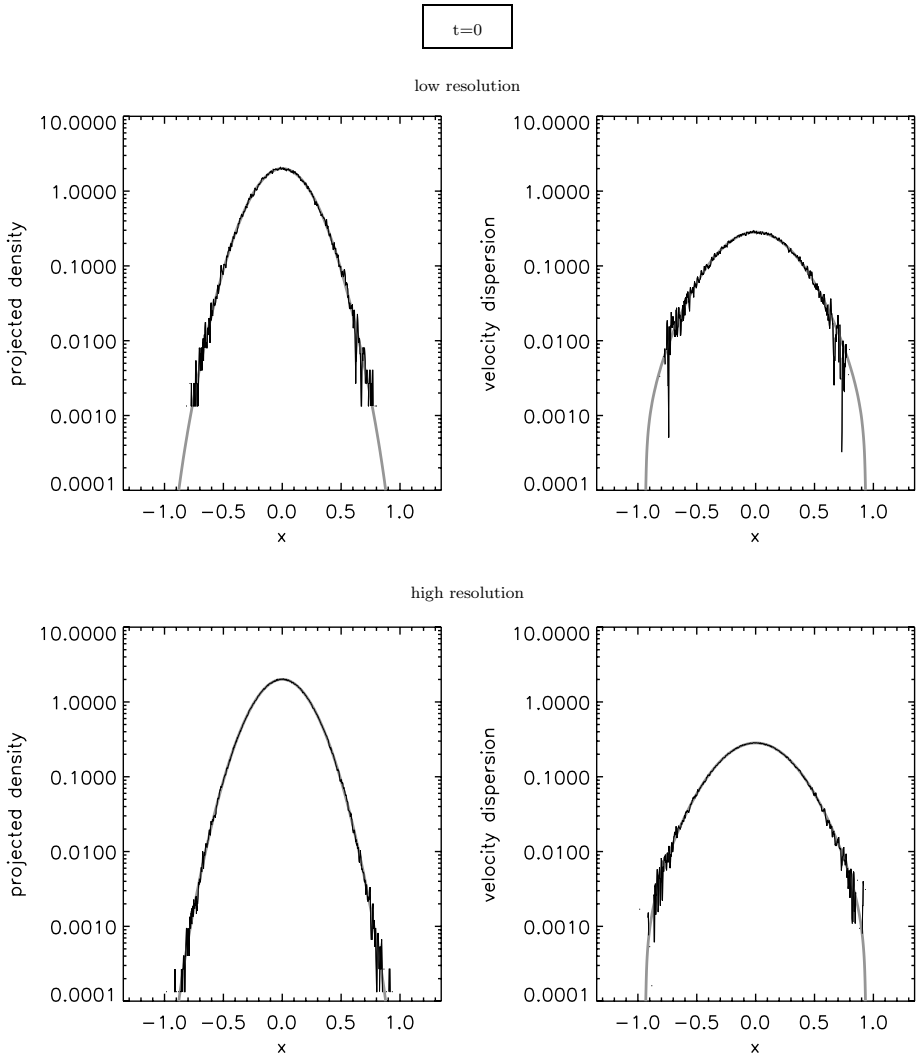

Figure 20. Mean projected density (left-hand panels) and velocity dispersion (right-hand panels), as functions of position, for the N-body simulations with Gaussian initial conditions, compared at various times, $t = 0$, 10, 40 and 100, to the result obtained with the cloud method. Each row of panels corresponds to low- and high-resolution N-body simulations, respectively. The N-body results are displayed as thin black curves, while the thick grey curve indicates the result obtained with the high-resolution cloud simulation. (Note that for $t = 0$, the thick grey curve is not exactly the same as in Fig. 13; it corresponds to the actual realization of the analytic initial conditions with the clouds.)

to a 5σ level for our Gaussian. In order to represent the initial conditions in phase space, we generate a set of particles with random positions according to a given density law. The time integrator is a simple leap-frog with constant time-step, $\mathrm{d}t = 0.0125$, comparable with the time-step chosen for our cloud simulations.

We performed two simulations: one at 'low resolution', with $N = 10^5$, a number of sampling units comparable to our high-resolution cloud simulation; and the other one at 'high resolution', with $N = 10^6$. Fig. 19 shows, similarly as Fig. 12, the phase-space distribution function at various stages for each simulation. To sample the density in phase space, we perform top-hat smoothing with a square window of size Δ_w equal to the high-resolution cloud simulation interspacing: $\Delta_w = \Delta_g = 0.005$. With this choice of Δ_w, we can perform a direct comparison between the N-body method and the cloud method, since the cloud interspacing basically defines the size of the smallest structures that the clouds can sample. Up to $t = 40$, the match with the cloud simulation is excellent, except of course the fact that the phase-space distribution function is more noisy. This visual match is confirmed by Fig. 20, which displays the projected density and the velocity dispersion at the same instants as in Fig. 19, where agreement between N-body and cloud methods is very good, even at $t = 100$, except in the tails, as expected.

At $t = 100$, the fine features in the phase-space distribution function are significantly distorted in the low-resolution simulation, probably due to the appearance of a resonance. This resonance seems to be also present in the high-resolution simulation, but to a much lesser extent. Even though we might expect this resonance to be of a physical nature (Touma, private communication), it should appear only at the microscopic level: for instance, we noticed that increasing the number of particles would considerably diminish its effects. Note that these effects are not so dramatic if one considers projected quantities such as density and velocity dispersion as illustrated by Fig. 19.

We already see here that our cloud method has two advantages. First, it preserves the smoothness and the features of the distribution function, even when f is small, which is not the case for the N-body approach: even with 10 times more sampling units, the N-body approach is rather noisy in the tails, as further illustrated by Fig. 20. Secondly, it performs actual coarse-grain dynamics, without propagating artificially possible resonances that should exist only at the microscopic level.

Note, however, that the N-body results can still be improved in different ways. For instance, instead of setting up initial conditions with a random distribution of particles, we could have reduced

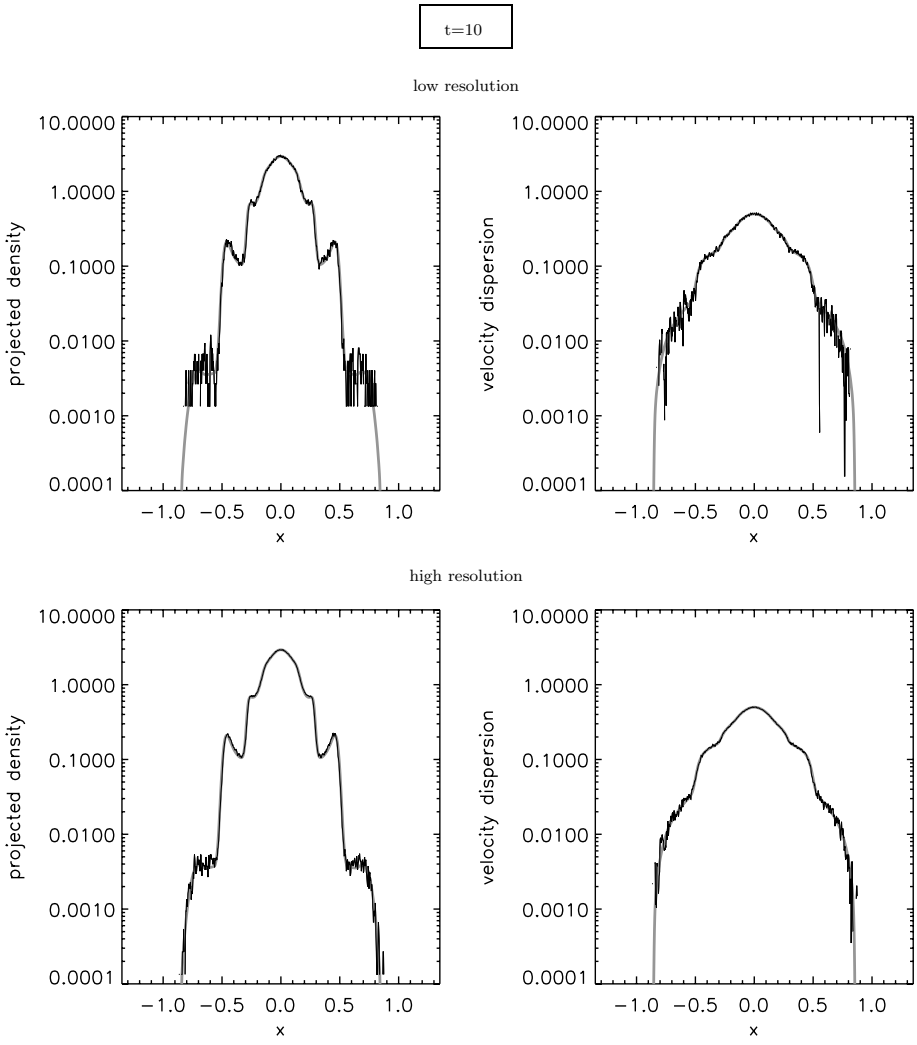

Figure 20 – *continued*

shot-noise effects by locating the particles on a regular pattern or on a glass deformed smoothly with the appropriate displacement field to match the initial distribution. We could as well improve the interpolation to compute the distribution function in phase space by using, e.g. Gaussian smoothing with the appropriate window size (which could be adaptive) or Voronoi tessellation (e.g. Arad, Dekel & Klypin 2004). We could as well transport the information relative to the initial value of f, and to avoid the appearance of the resonance noticed above, we could improve the calculation of the force to make it smooth enough. With all these potential improvements, there is no doubt the N-body method would perform much better in terms of accuracy, but at a significant computational cost.

It would be beyond the scope of this paper to compare in detail the computational cost of our method with an 'improved' N-body, especially since we are treating here the simple two-dimensional phase-space case: it will be more relevant to discuss this matter when the method will be applied to realistic situations in six-dimensional phase space. However, we can give here a qualitative comparison. In two-dimensional phase space, compared to standard N-body, our method is of course more costly in memory: two parameters corresponding to the shape of the clouds have to be transported and the force must be supplemented with its slope. It is also more costly in CPU: the slope of the force has to be estimated and the Lagrange remap has to be performed. This last step is rather expensive in the tree-code version of our code but can be greatly speeded up in the CM version, by using the fast convolution method explained in Appendix A2. In terms of CPU time, the CM version compares fairly well with our PM code: it is typically an order of magnitude slower for the same number of sampling units. However, as shown above, the gain in accuracy and smoothness is considerable compared to the overcost in CPU.

6 ADAPTIVE REFINEMENT

In the simulations with Gaussian and top-hat initial conditions studied in Section 4, the complexity of the phase-space distribution function augments with time, due to the well-known effect of rolling up. In the top-hat case, details are significant only in the external ring. This calls for local refinement, i.e. for increasing resolution only where needed: in principle this should reduce significantly the cost of the simulation. The gain obtained from local refinement in two-dimensional phase space might of course be quite questionable. For instance, such a gain is expected to be rather small in the Gaussian case, where complexity appears everywhere in the computing domain with approximately the same level of detail. However, refinement is expected to be much more relevant in higher numbers

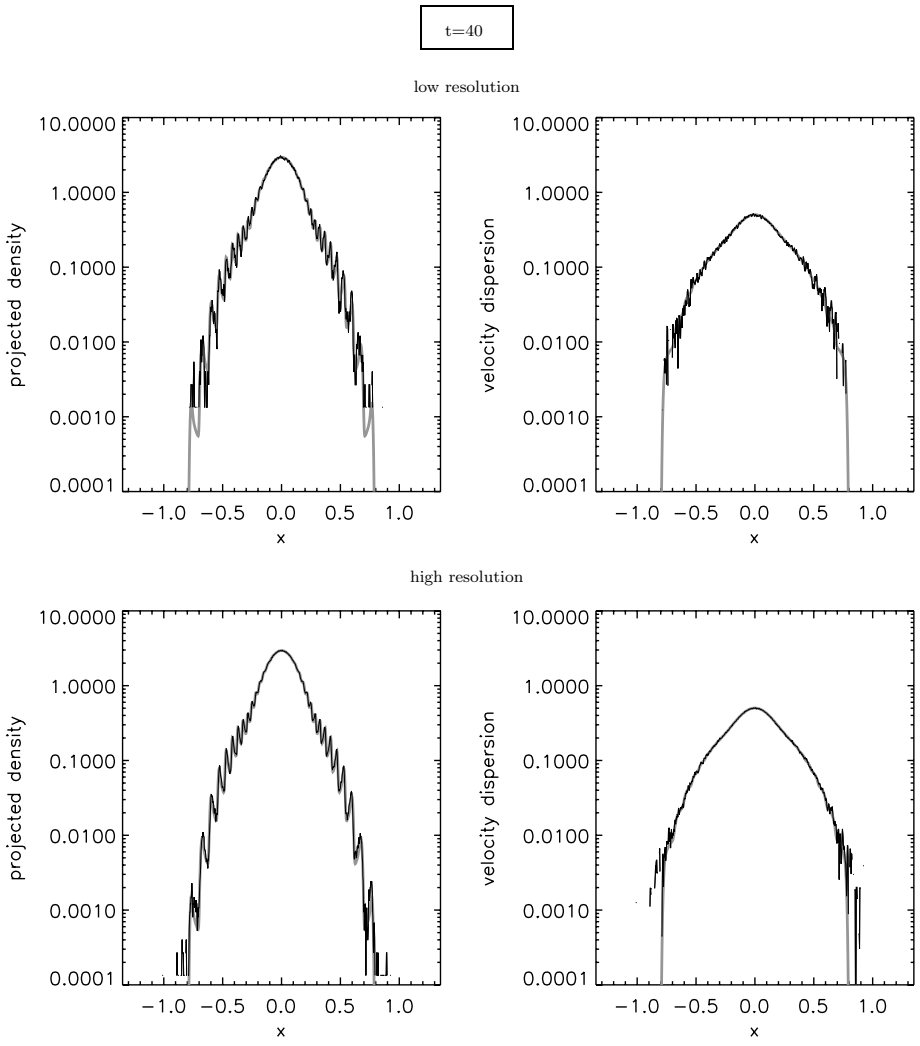

Figure 20 – *continued*

of dimensions, especially in the cosmological case, where only a very small fraction of phase space is occupied.

Here, we examine a simple approach based on standard adaptive refinement tree (ART) methods. Our goal is to demonstrate only that refinement is possible with the cloud method. This section is organized as follows. In Section 6.1, we give a sketch of the method. Technical details are discussed in Appendix B. In Section 6.2, we show how the method performs for the top-hat initial conditions used in Section 4.3: we start from a low-resolution initial set-up with $\Delta_x = \Delta_v = 0.02$, allow for two levels of refinement, and compare the results obtained to the high-resolution simulations of Section 4.3, with $\Delta_x = \Delta_v = 0.005$.

6.1 Method of refinement

In the practical implementation that we test on the tree-code, the phase space is decomposed hierarchically on a quad-tree at the moment of remap. Each cell of the quad-tree is associated to a cloud. If needed, the cell is split equally into four subcells corresponding to four subclouds. The process is performed as long as necessary, i.e. until the sizes of the corresponding (sub)clouds obey some criteria based on local properties of the phase-space distribution function.

At each successive level of refinement, we use the van Citter algorithm to reconstruct the distribution function. We first start from the coarse level, where we reconstruct f the same way as for the fixed-resolution code, as described in Section 3.1.2. We then consider the set of clouds corresponding to the first level of refinement as residues to reconstruct f more accurately by applying the van Citter algorithm again on these clouds, and so on, until the last level of refinement.

We consider two criteria of refinement. The first one is based on the measurement of local curvature of the phase-space distribution function: local curvature is indeed a key quantity for preserving details in the distribution function: the cloud size should be always small compared to the local curvature radius. The second criterion is based on convergence of the reconstruction at successive refinement levels: clouds where f is poorly reconstructed are refined. We shall see that in practice both curvature and convergence criteria give similar refinement structure.

In our implementation of refinement, even when split into smaller subclouds, clouds at a given refinement level are thus kept since

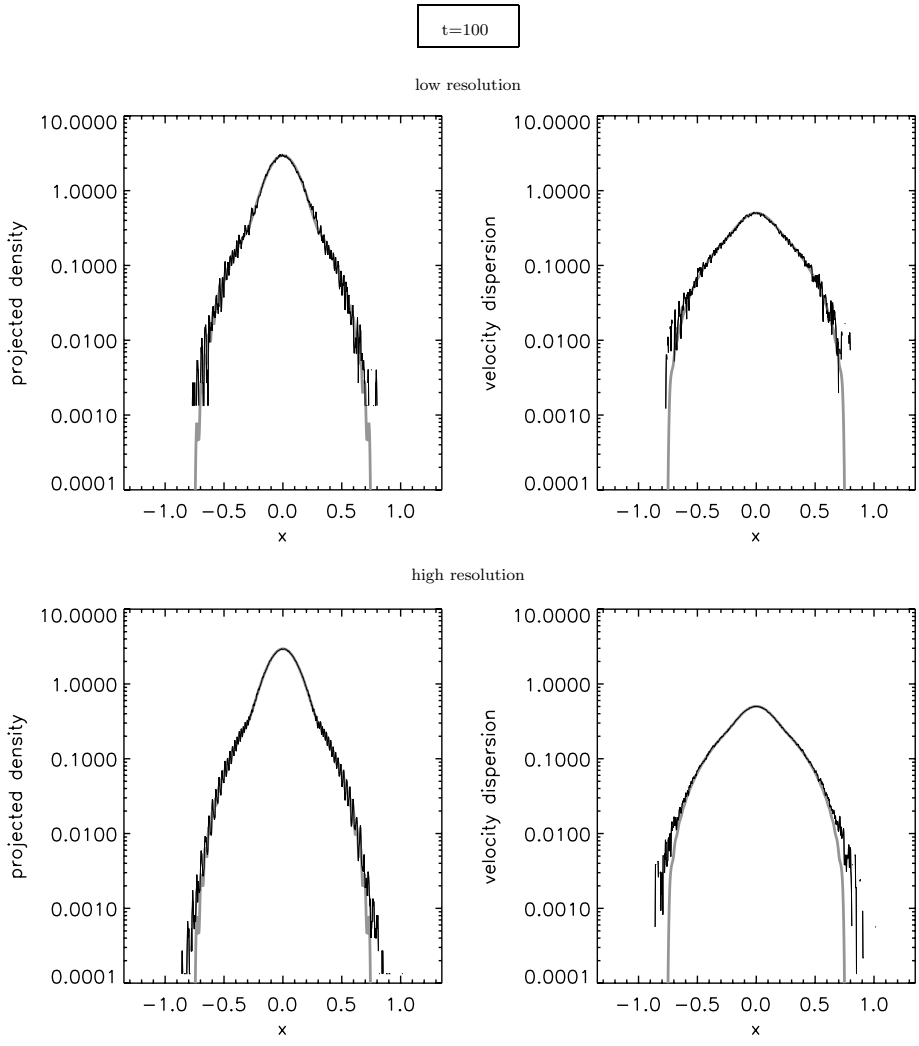

Figure 20 – *continued*

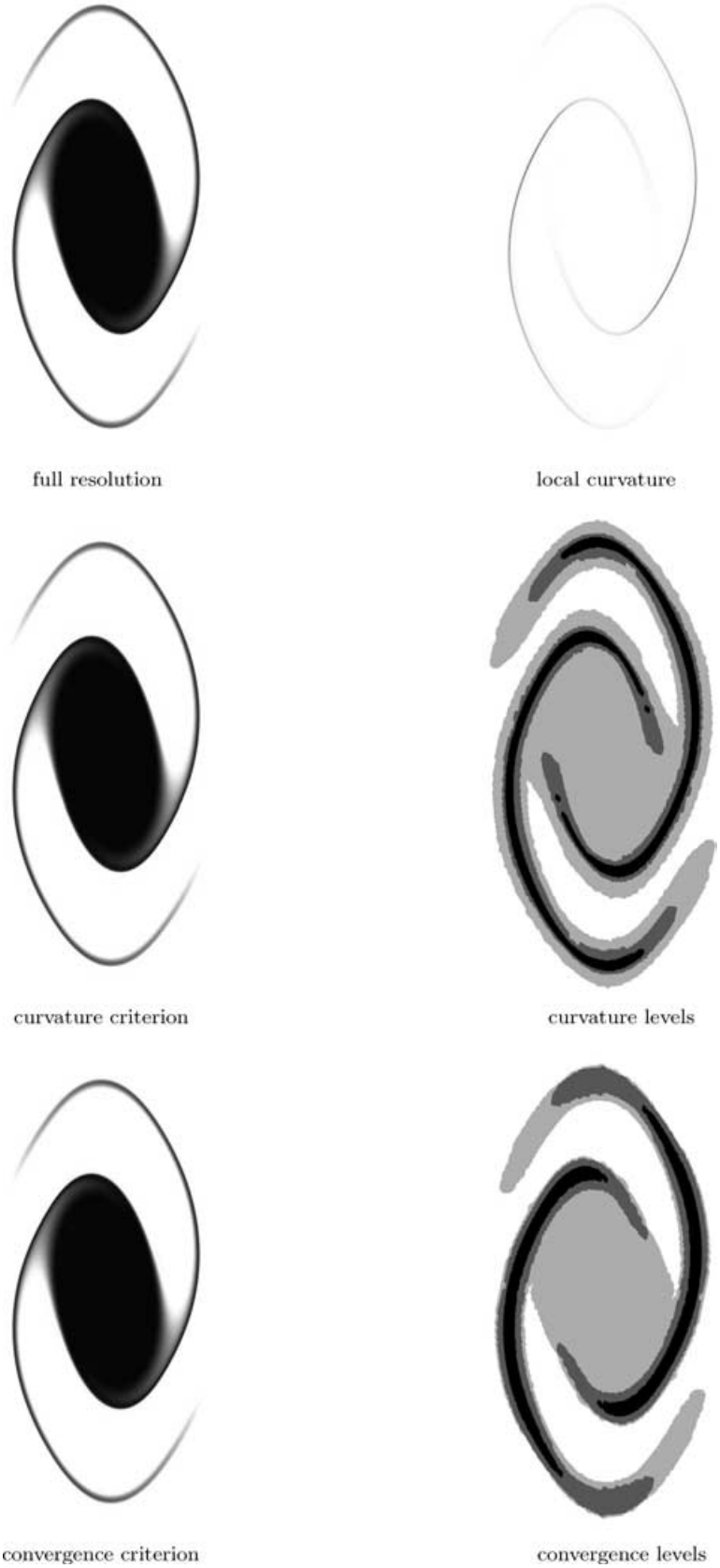

Figure 21. Top-hat simulations at $t = 10$. Left-hand panels: phase-space distribution function for the full-resolution simulation (top), the simulation with refinement based on local curvature (middle) and the simulation with refinement based on local convergence (bottom). Top right-hand panel: the local curvature measured in the simulation with refinement based on local curvature, as explained in the text of Section 6.2. Two bottom right-hand panels: levels of refinement in the simulations with local curvature criterion (middle) and local convergence criterion (bottom). Darker regions corresponds to higher level.

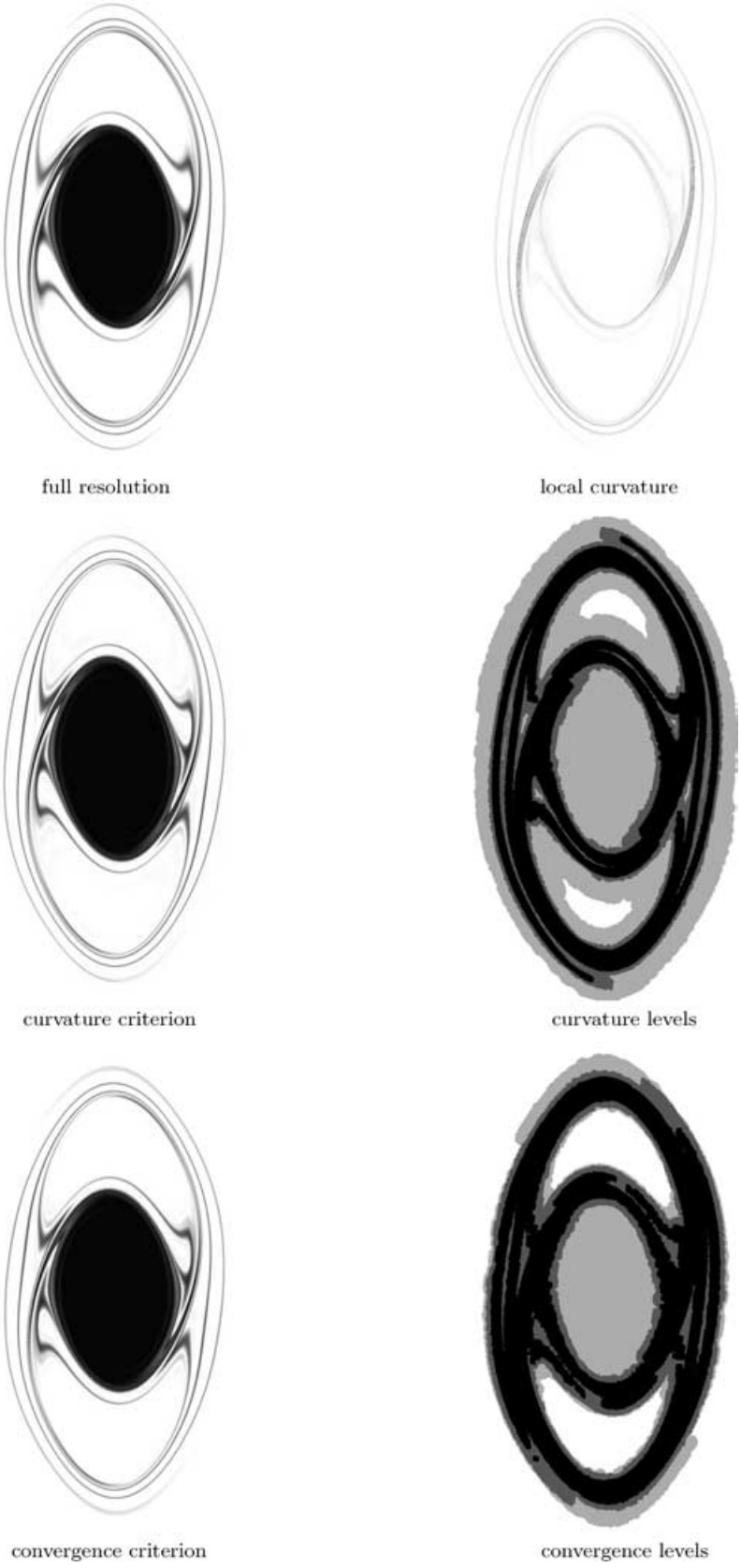

Figure 22. Same as in Fig. 21, but for $t = 40$.

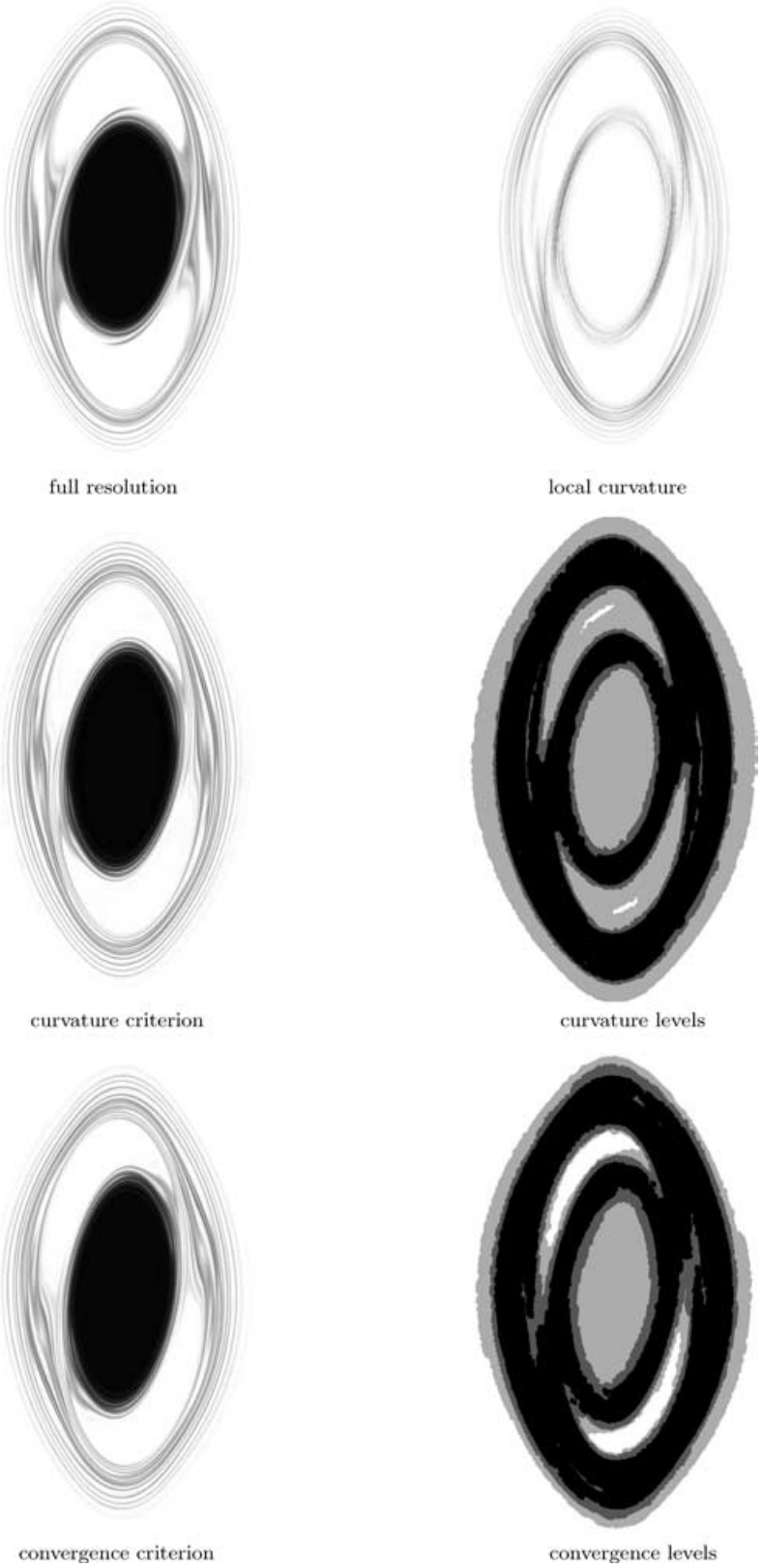

Figure 23. Same as in Fig. 21, but for $t = 100$.

their subcloud counterparts are considered as residues. In principle, it would be advisable from the dynamical point of view to remove these clouds from the hierarchy, i.e. to set their mass to zero. Indeed, because of the structures in the gravitational potential brought by the small clouds, the quadratic approximation is expected at some point to become invalid for the larger clouds. We notice as well that one of our refinement criteria should rely as well on variations of the projected density, which account for deviations from the quadratic

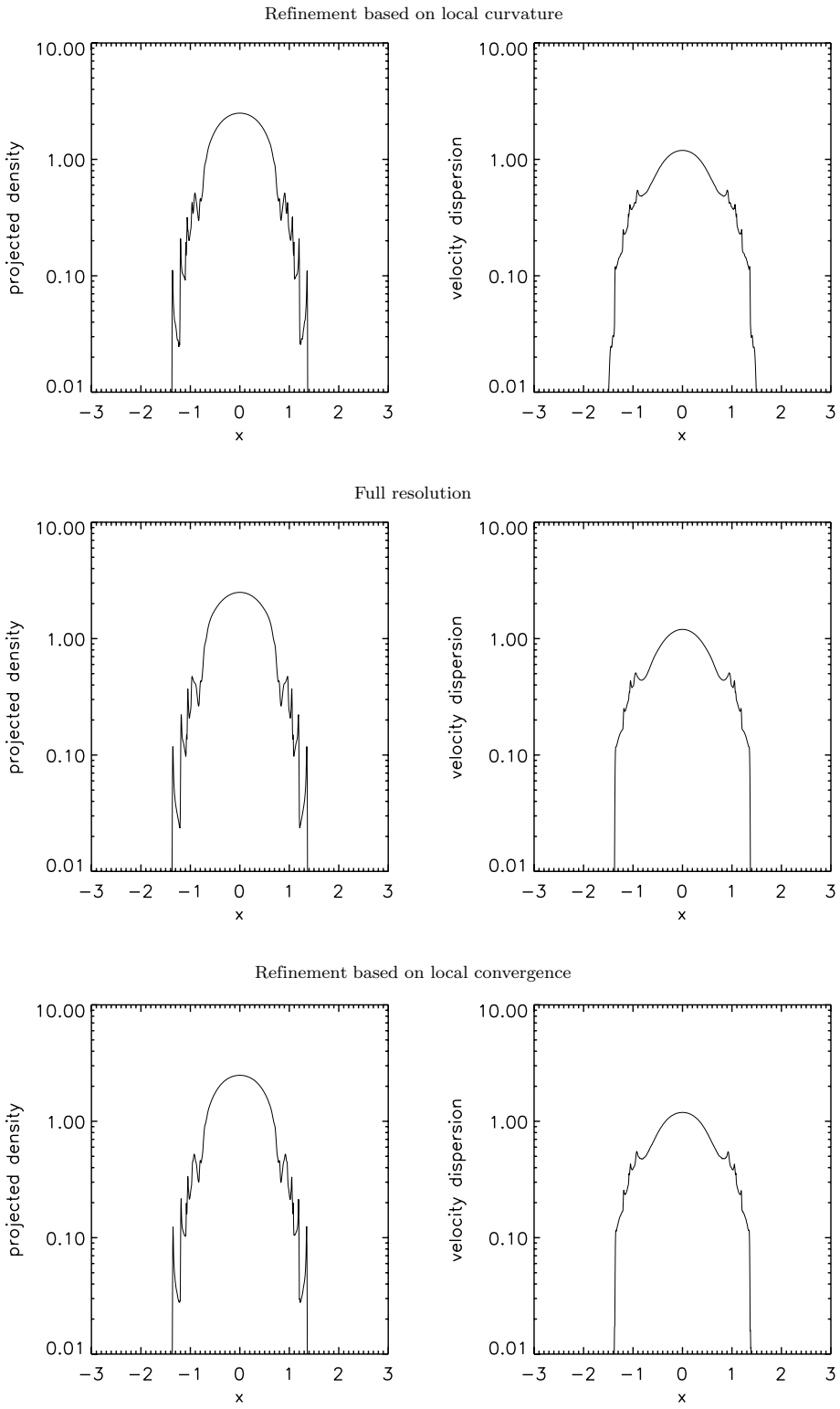

Figure 24. Mean projected density (left-hand panels) and velocity dispersion (right-hand panels), as functions of position, for top-hat simulations with refinement based on local curvature (top row) and on local convergence (bottom row), compared to full resolution (middle row), at $t = 40$.

behaviour of the potential. As our goal is just to demonstrate that refinement is possible with our method, we decided in the present work to put aside removal of coarser level clouds and refinement based on projected density. As it is, thus, our refinement procedure is quite improvable and will work only if the gravitational potential remains sufficiently quadratic at the coarse level, which should hopefully be the case if the number of refinement levels is not too large.

6.2 Example: top-hat initial conditions

To check how our refinement procedure performs, we again ran simulations with the same top-hat initial conditions as in Section 4.3,

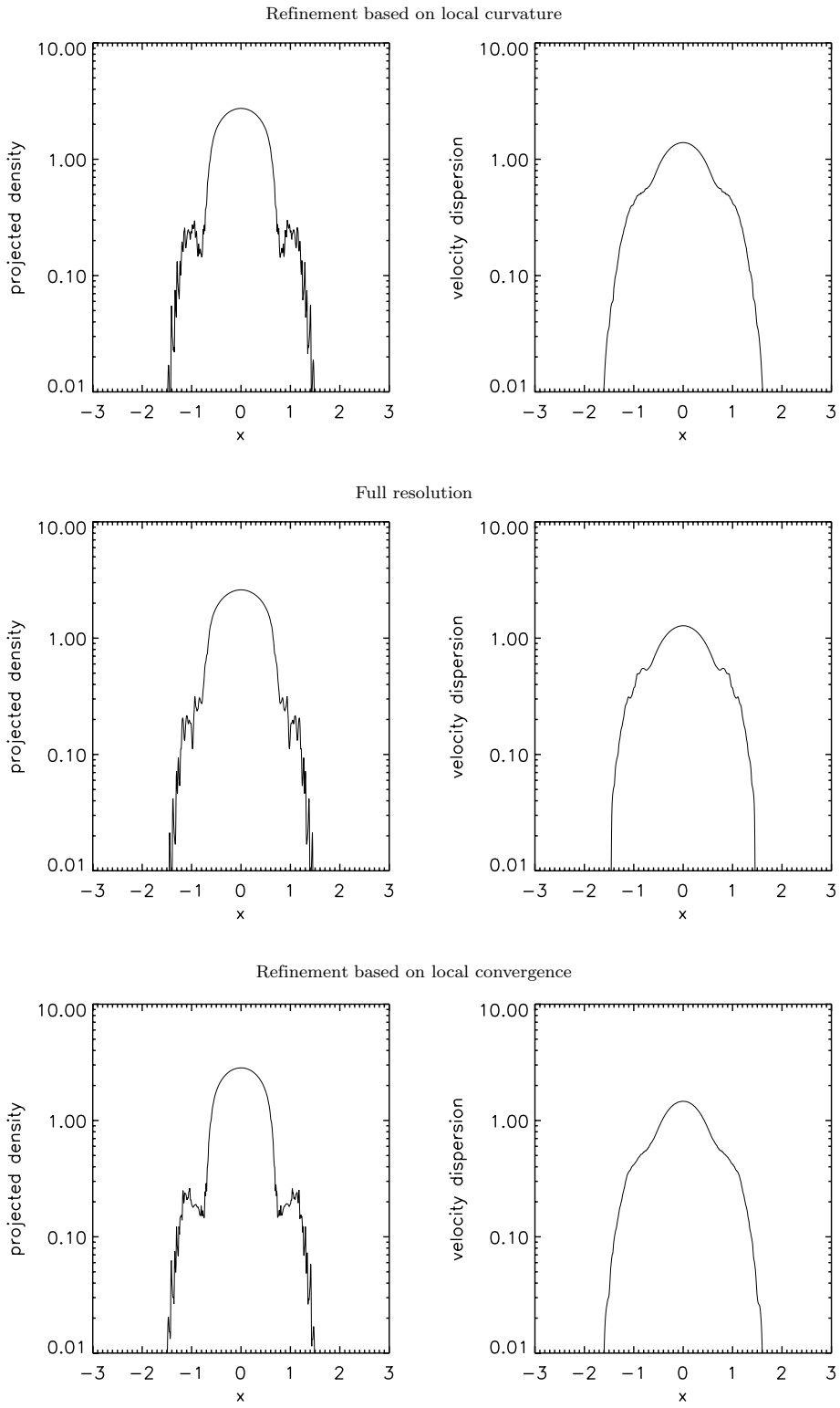

Figure 25. Same as Fig. 24, but for $t = 100$.

but starting from low-resolution initial conditions with $\Delta_g = 0.02$ and allowing refinement until high resolution is reached, $\Delta_g = 0.005$. This represents a significant increase in mass resolution, a factor 16, corresponding in total to a coarse level plus two levels of refinement. We performed two simulations, based on local curvature and local convergence criteria, respectively. All the parameters defining the simulations are the same as in Section 4, except for those depending on refinement, which are given in Appendices B1 and B2. There is also a difference in the choice of $\delta_{\rm min}$ (see Section 3.1.2) that we set to $\delta_{\rm min} = 0.0005$ for simulations based on adaptive refinement with the local convergence criterion.

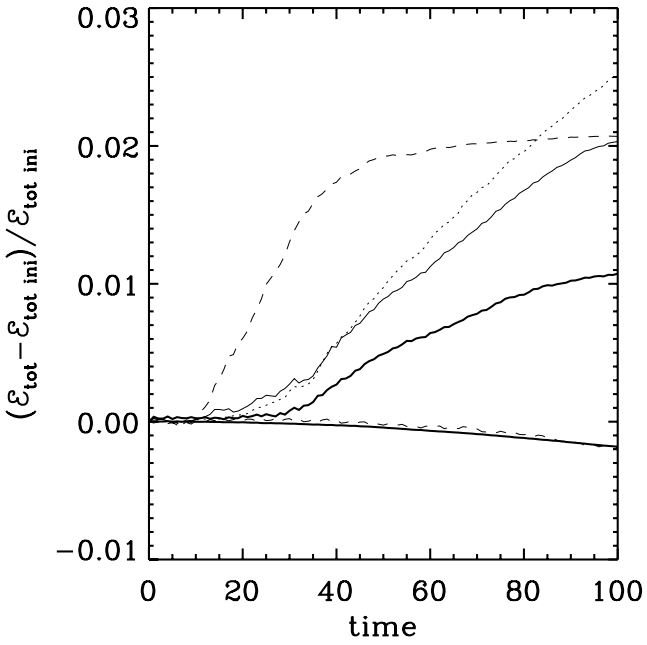

Figure 26. Energy conservation in top-hat simulations with refinement, compared to fixed-resolution simulations. The solid and dotted thin curves correspond to simulations with refinement based on local curvature and local convergence, respectively. The lower thick curve and nearly superposed dashed one correspond, respectively, to the high- and low-resolution simulations of Section 4.3. These simulations use Lucy deconvolution. The upper thick and dashed curves correspond to similar simulations, but were performed using van Citter reconstruction.

Figs 21, 22 and 23 show the phase-space distribution function at various times, $t = 10$, 40 and 100. The refined simulations seem to reproduce rather well the results of the full-resolution simulation, up to $t = 40$. For $t = 100$, the refined simulations clearly differ from the full-resolution one, even though they present the same level of detail. The two lowest right-hand panels on each of the figures show the refinement levels. As expected, the result obtained from the local curvature criterion is very similar to that obtained from the local convergence criterion. For reference, the upper right-hand panel shows an estimate of the local curvature, more exactly the quantity $\mathcal{C} = \max(|\lambda_1|, |\lambda_2|)$ present in equation (B4) where λ_1 and λ_2 are the eigenvalues of the Hessian of the phase-space distribution function. Note that $\mathcal{C}$ is quite noisy despite our sophisticated weighting scheme to compute f and its derivatives, as it still captures some small-scales defects, but it seems to be determined accurately enough to set up refinement correctly based on local curvature.

Figs 24 and 25 show the projected density and the velocity dispersion for the snapshots $t = 40$ and $t = 100$. Careful examination confirms the visual impression from Figs 22 and 23: the refined simulations reproduce well the fine features of the full-resolution one for $t = 40$, but not for $t = 100$.

Finally, Fig. 26 examines energy conservation for various simulations. Refined simulations present the same behaviour, whether a local curvature or local convergence criterion is used: excellent energy conservation up to $t \simeq 40$, then a nearly linear increase of energy with time with a significant but still reasonable final violation of energy conservation of the order of 2 per cent. The curves corresponding to the high- and low-resolution simulations of Section 4.3, performed with Lucy deconvolution, are displayed for reference, but it would be more fair to make the comparison with fixed-resolutions simulations performed with van Citter. Indeed, in our current implementation, simulations using van Citter deconvolution conserve energy less well than those using Lucy: in Fig. 26, the high-resolution simulation using van Citter presents similar behaviour to our simulations with refinement, except that the final violation of energy conservation is twice smaller. The low-resolution simulation does significantly worse. First it conserves energy well, up to $t \simeq 10$, then energy increases rather fast up to $t \simeq 40$ where it reaches a plateau corresponding to a global violation of energy conservation of the order of 2 per cent.

This behaviour is related to the fact that the positivity of the distribution function is not guaranteed in the van Citter algorithm and that we perform a truncation of the tails of f: if f is smaller than $\delta_{\rm min}$, it is set to zero. If this truncation, which is meant to keep the computing domain finite and f as positive as possible, was not present, energy conservation would be by construction much better. The cut-off of the negative contributions of f implies that energy increases with time. The effect is all the stronger since resolution is low. Refinement is expected to improve energy conservation compared to low resolution, at least during some length of time, but since it keeps coarse and intermediary levels of refinement, it is not expected to do as well as the full-resolution simulation, hence the result observed in Fig. 26.

However, a 2 per cent violation of energy conservation is probably not enough to explain the significant disagreement at $t = 100$ between simulations with refinement and the full-resolution one. There is indeed at least one other effect intervening, as discussed in the last paragraph of Section 6.1: namely that at the coarser levels, the local quadratic nature of the potential is likely to be violated during run-time. This can produce cumulative effects that show up only after a large number of orbits.

Despite the limitations of the current implementation, we think we have demonstrated here that the cloud method is compatible with adaptive refinement.

7 PERSPECTIVES AND EXTENSION TO A HIGHER NUMBER OF DIMENSIONS

We demonstrated the ability of the cloud method to solve the Vlasov–Poisson system in one dimension. An interesting feature of this method is that its structure allows a simple generalization to the n-dimensional case. We will show in the next section that the n-dimensional cloud equations can be solved quite easily, as a natural generalization of the one-dimensional equations. It is also quite obvious that the tree-code method that we used to solve the Poisson equation can be extended to a higher number of dimensions.

7.1 *n*-dimensional equations

Provided that we keep in mind the results obtained in one dimension, the generalization of the cloud equation to n dimensions is straightforward. In the one-dimensional case, it was shown that a general solution cloud can be written as a function of second-order polynomials in $(x_1, \ldots, x_n, v_1, \ldots, v_n)$. We will adopt this result and show that a closed system of equations can be obtained for an n-dimensional cloud. To simplify the writing of the equations, we define the vector $\boldsymbol{u}$ such that:

$$u_i = \begin{cases} x_i & 0 < i \leqslant n, \\ v_i & n < i \leqslant 2n. \end{cases} \tag{91}$$

Using this notation, we are now able to write the generalized Vlasov equation in Lagrangian coordinates for a cloud of density $f(\boldsymbol{u}, t)$

evolving in the potential $\phi(\boldsymbol{u}, t)$:

$$\frac{\partial}{\partial t} f(\boldsymbol{u}, t) + \frac{\partial}{\partial u_i} f(\boldsymbol{u}, t)\, u_{i+n}\, \eta(i) - \frac{\partial}{\partial u_i}\phi(\boldsymbol{u}, t)\, \frac{\partial}{\partial u_{i+n}} f(\boldsymbol{u}, t)\, \eta(i) = 0, \tag{92}$$

with the following definition for the function η:

$$\eta(i) = \begin{cases} 1 & 0 < i \leqslant n, \\ 0 & i < 1 \text{ or } i > n. \end{cases} \tag{93}$$

We will look for a general solution in the case of a quadratic potential. An extrapolation of the one-dimensional case suggests that the general cloud equation can be written as a function of a quadratic polynomial. In this case:

$$\phi(\boldsymbol{u}, t) = \alpha_{ij}\, \eta(i)\eta(j)\, u_i u_j, \tag{94}$$

$$f(\boldsymbol{u}, t) = F(\lambda_{ij}\, u_i u_j). \tag{95}$$

Note that the coefficients α_{ij} and λ_{ij} depend on time, in general, $\alpha_{ij} \equiv \alpha_{ij}(t)$ and $\lambda_{ij} \equiv \lambda_{ij}(t)$.

By inserting this expression for $f(\boldsymbol{u}, t)$ in the Vlasov equation, equation (92), we will obtain a closed system of differential equations for the coefficients $\lambda_{ij}(t)$. We will now illustrate the relevant calculations step by step. By taking each term in the Vlasov equation from left to right we have:

First term in the Vlasov equation:

$$\frac{\partial}{\partial t} f(\boldsymbol{u}, t) = \frac{\partial \lambda_{ij}}{\partial t}\, u_i u_j. \tag{96}$$

Second term in the Vlasov equation:

$$\frac{\partial}{\partial u_i} f(\boldsymbol{u}, t)\eta(i) = 2\lambda_{ij}\eta(i)\, u_j. \tag{97}$$

Thus,

$$\frac{\partial}{\partial u_i} f(\boldsymbol{u}, t)\, u_{i+n}\eta(i) = 2\lambda_{ij}\, \eta(i)\, u_j u_{i+n}, \tag{98}$$

which can be rewritten:

$$\frac{\partial}{\partial u_i} f(\boldsymbol{u}, t)\, u_{i+n}\eta(i) = 2\lambda_{(k-n)j}\, \eta(k-n)\, u_j u_k. \tag{99}$$

The last term of the Vlasov equation can be evaluated using the same method. We arrive at the following result:

$$\frac{\partial}{\partial u_i}\phi(\boldsymbol{x}, t)\frac{\partial}{\partial u_{i+n}} f(\boldsymbol{u}, t)\, \eta(i) = 4\phi_{ik}\, \lambda_{(k+n)j}\, \eta(i)\eta(k)\, u_j u_k. \tag{100}$$

We are now ready to tackle the Vlasov equation itself. We have to write that each coefficient of the second-order polynomial is zero. For each second-order term $u_i u_j$ we have to consider the contribution from the pair of indices (i, j) and (j, i). In the case $i = j$, there is only one contributor, but we will still add the symmetric term; this will result in a factor of 2 on the left side, and since the right side is equal to 0 it does not change the equation. Thus by adding the contribution of each term in equations (96), (99) and (100), and taking into account the symmetric term, we find that:

$$\frac{\partial \lambda_{ij}}{\partial t} + \lambda_{(i-n)j}\, \eta(i-n) + \lambda_{(j-n)i}\, \eta(j-n) - 2\phi_{ki}\, \lambda_{(k+n)j}\, \eta(k)\eta(i) - 2\phi_{kj}\, \lambda_{(k+n)i}\, \eta(k)\eta(j) = 0. \tag{101}$$

7.2 Applications of the method

Considering the current capabilities of the largest computers it seems clear that the two-dimensional case could be undertaken rapidly, and that in astrophysics it would offer a new look at a few interesting problems, like for instance the dynamics of galactic discs. Due to the particular interest of the two-dimensional case we present the detailed cloud equations in Appendix C. Due to the high dimensionality of phase space in the three-dimensional case, it seems that trying to integrate the Vlasov–Poisson system directly using the cloud method may be too costly. However, it is clear that a most interesting case, namely the cold dark matter model in cosmology, may not require the same amount of resources and could be solved using an appropriate version of the cloud method. It is important to notice that in this case the density in the six-dimensional phase space is adequately represented by the extended folding of a three-dimensional sheet having a nearly constant density. In particular, this case may not require the sophisticated Lucy deconvolution scheme and the general remapping technique.

ACKNOWLEDGMENTS

We thank B. Paczyński, R. Teyssier, J. Touma and S. Tremaine for useful discussions. We also thank V. Pislar for careful reading of the paper and double checking of analytic results. This work was performed within the Numerical Investigations in Cosmology (NIC) group as a task of the HORIZON project.

REFERENCES

Arad I., Dekel A., Klypin A., 2004, MNRAS, 353, 15
Arber T. D., Vann G. L., 2002, J. Comput. Phys., 180, 339
Besse N., Sonnendrücker E., 2003, J. Comput. Phys., 191, 341
Channell P. J., 1995, Ann. New York Acad. Sci., 751, 152
Cheng C. Z., Knorr G., 1976, J. Comput. Phys., 22, 330
Cuperman S., Harten A., Lecar M., 1971, Ap&SS, 13, 411
DePackh D. C., 1962, J. Electr. Contr., 13, 417
Filbet F., Sonnendrücker E., Bertrand P., 2001, J. Comput. Phys., 172, 166
Fujiwara T., 1981, PASJ, 33, 513
Gagné R. R., Shoucri M., 1977, J. Comput. Phys., 27, 315
Hockney R. W., Eastwood J. W., 1988, Computer Simulation using Particles. IOP Publishing, Bristol
Hohl F., Feix M. R., 1967, ApJ, 147, 1164
Janin G., 1971, A&A, 11, 188
Klimas A. J., 1987, J. Comput. Phys., 68, 202
Klimas A. J., Farrell W. M., 1994, J. Comput. Phys., 110, 150
Merrall T. E. C., Henriksen R. N., 2003, ApJ, 595, 43
Nishida M. T., Yoshizawa M., Watanabe Y., Inagaki S., Kato S., 1981, PASJ, 33, 567
Rasio F. R., Shapiro S. L., Teukolsky S. A., 1989, ApJ, 344, 146
Roberts K. V., Berk H. L., 1967, Phys. Rev. Lett., 19, 297
Shlosman I., Hoffman Y., Shaviv G., 1979, MNRAS, 189, 723
Sonnendrücker E., Roche J., Bertrand P., Ghizzo A., 1999, J. Comput. Phys., 149, 201
Spitzer L., 1942, ApJ, 95, 329
Syer D., Tremaine S., 1995, MNRAS, 276, 467
Watanabe Y., Inagaki S., Nishida M. T., Tanaka Y. D., Kato S., 1981, PASJ, 33, 541
White R. L., 1986, in McMillan S., Hut P., eds, Use of Supercomputers in Stellar Dynamics. Springer, Berlin, p. 167
Zaki S. I., Gardner L. R. T., Boyd T. J. M., 1988, J. Comput. Phys., 79, 184

APPENDIX A: SAMPLING OF THE DISTRIBUTION FUNCTION FROM A SET OF CLOUDS

A1 Arbitrary set of clouds on a regular grid

Assume we want to compute $f_{i,j} = f(x_i, v_j)$ where (x_i, v_j) are sampling points on a square grid of spacing Δ, for an ensemble of clouds with arbitrary axis ratio and orientation, using either equation (59) or equation (66). Since the clouds are rather extended and overlap significantly, the calculation of $f_{i,j}$, is expected to be rather CPU-time consuming, in general, but simple. The natural way of performing it is simply to project each cloud independently on the grid. The only difficulty is to find the subsample of grid points where a given cloud contributes. To do that, one can estimate the projected size of the cloud on the x- and v-axes:

$$R_{\max,x} = \frac{R_{\max}}{R}\sqrt{\frac{\beta_2}{2C}}, \quad R_{\max,v} = \frac{R_{\max}}{R}\sqrt{\frac{\beta_0}{2C}}, \tag{A1}$$

where β_0 and β_2 are given by equations (22) and (24), respectively. This defines the rectangle where the ellipse corresponding to the cloud is inscribed. To speed up the calculation, one can find, for fixed values of $x \in [-R_{\max,x}, R_{\max,x}]$ in the cloud coordinate frame, the segment $[v_1, v_2]$ given by the intersection between the ellipse covered by the cloud and the vertical line of abscissa x, which we do not feel necessary to write here for simplicity.

A2 Round set of clouds on a grid: fast convolution algorithm

When one considers a round set of clouds set up on a regular grid, it is possible to use a simple trick to speed up the calculation of $f_{i,j}$ compared to the simple algorithm described in Section A1, by noticing that the convolution of a function by a Gaussian can be factorized in that case:

$$\begin{aligned}\int \mathrm{d}x^*\,\mathrm{d}v^*\; f(x+x^*, v+v^*)\,\exp\left[-\frac{1}{2}\left(\frac{x^{*2}}{R_x^2}+\frac{v^{*2}}{R_v^2}\right)\right] \\ = \int \mathrm{d}x^*\,\exp\left(-\frac{1}{2}\frac{x^{*2}}{R_x^2}\right) \\ \times \int dv\; f(x+x^*, v+v^*)\,\exp\left(-\frac{1}{2}\frac{v^{*2}}{R_v^2}\right).\end{aligned} \tag{A2}$$

In practice, this means the following. Assume for simplicity that the centre, $(x_{\mathrm{G}}^n, v_{\mathrm{G}}^n)$, of each cloud, n, coincides exactly with a grid sampling point (x_{i_n}, v_{j_n}), and furthermore that cloud interspacing, Δ_{g}, is a multiple of the target mesh interspacing Δ. With these assumptions, first create an array $f_{i,j}^{**}$, which is zero everywhere, except at cloud positions

$$f_{i_n,j_n}^{**} = M_n/V, \tag{A3}$$

where M_n is the mass of the cloud n, and $V = \Delta_{\mathrm{g}}^2$. Then perform an operation equivalent to convolution in velocity space. For doing that, it is just necessary to propagate vertically the initial values f_{i_n,j_n}^{**} using the weights given by function $G(x, y)$, on the columns of the array $f_{i,j}^{**}$ which contains non-zero elements:

$$f_{i_n,j}^{*} = \sum_{m,|v_j - v_{\mathrm{G}}^m| \leqslant R_{\max}} f_{i_n,j_m}^{**} G\left(0, v_j - v_{\mathrm{G}}^m\right). \tag{A4}$$

The last operation consists of convolving along x-axis:

$$f_{i,j} = \frac{1}{G(0,0)} \sum_{m,|x_j - x_{\mathrm{G}}^m| \leqslant R_{\max}} f_{i_m,j}^{*} G\left(x_j - x_{\mathrm{G}}^n, 0\right). \tag{A5}$$

Here we used the separability of the function $G(x, y)$: $G(x, y) = G(x, 0)\, G(0, y)/G(0, 0)$. We made the approximation, to speed up the calculation, that the truncation is not $x^2 + v^2 \leqslant R_{\max}^2$, but rather $|x| \leqslant R_{\max}$ and $|v| \leqslant R_{\max}$, but this should not have any significant consequence if the value of $R_{\max}$ is large enough, e.g. $R_{\max} = 4R$ as used in this paper. For simplifying the argument, we introduced the arrays $f_{i,j}^{**}$ and $f_{i,j}^{*}$ with the same size as the target mesh size, but one can see that only f^* is necessary, and it can be reduced to an array of dimension $(N_{\mathrm{c},x}, N_{\mathrm{g},v})$, where $N_{\mathrm{c},x}$ and $N_{\mathrm{g},v}$ are the size of the cloud mesh along the x-axis and of the target mesh along the v-axis, respectively. Typically, the first step of the calculation, equation (A4), takes $2\, N_{\mathrm{tot}} R_{\max}/\Delta$ operations, where N_{tot} is the total number of clouds, while the second step, equation (A5), takes $2\, N_{\mathrm{sites}} R_{\max}/\Delta$, where N_{sites} is the total number of target grid sites. The total number of operations, $(N_{\mathrm{tot}} + N_{\mathrm{sites}})\, 2R_{\max}/\Delta$, is to be compared to $\pi\, N_{\mathrm{sites}}(R_{\max}/\Delta)^2$, as expected from the method presented in Section A1, a tremendous gain in time. Indeed, if we use equations (A4) and (A5) to compute $\tilde{f}_i$ in the Lucy or van Citter algorithms, we see that we gain a factor of order $(4/\pi)(R_{\max}/\Delta)^2 \simeq 10$ for the deconvolution and our parameter choice ($R_{\max} = 4R$, $R = \Delta/\sqrt{2}$). As a result, we used this fast convolution method in the CM code to speed up reconstruction and the method described in Section A1 in other cases, when the clouds have arbitrary shape and orientation.

In principle, we could have used this efficient implementation as well for our tree-code. However, this program is designed for adaptive mesh refinement (Section 6). It would be rather involved, algorithmically, although possible, to adapt this method to an unstructured grid such as the one used in our refinement procedure. Instead, for all the simulations presented in this paper, which are done with the tree-code, we used the very general but much slower method described below, at a significant cost of CPU time.

Note finally that equations (A4) and (A5) apply to equation (59), but the method can easily be generalized to equation (66).

A3 Arbitrary set of clouds at an arbitrary point: the quad-tree algorithm

In general, if one aims to estimate $f(x, v)$ at an arbitrary point of phase space, it is necessary to find rapidly the clouds contributing to this point. There is a very general and standard method of doing that, based on hierarchical decomposition of phase space on a quad-tree structure, until there is zero or one cloud centre per cell of this tree. The list of clouds contributing to point (x, v) is constructed by walking into this tree, from root to leaves (i.e. going through the structure of the tree, from the largest to the smallest structures). To build the tree properly, one has take into account the extension of the clouds. While the tree is constructed, for each cell it contains, defined by some coordinate range, $[(x_1, v_1), (x_2, v_2)]$, one computes the effective potential range $C_{\mathrm{eff}} \equiv [(x_{\mathrm{eff},1}, v_{\mathrm{eff},1}), (x_{\mathrm{eff},2}, v_{\mathrm{eff},2})]$, where

$$x_{\mathrm{eff},1} = \min_{\text{clouds } i \text{ in the cell}} \left(x_{\mathrm{G}}^i - R_{\max,x}^i\right), \tag{A6}$$

$$x_{\mathrm{eff},2} = \max_{\text{clouds } i \text{ in the cell}} \left(x_{\mathrm{G}}^i + R_{\max,x}^i\right), \tag{A7}$$

$$v_{\mathrm{eff},1} = \min_{\text{clouds } i \text{ in the cell}} \left(v_{\mathrm{G}}^i - R_{\max,v}^i\right), \tag{A8}$$

$$v_{\mathrm{eff},2} = \max_{\text{clouds } i \text{ in the cell}} \left(v_{\mathrm{G}}^i + R_{\max,v}^i\right). \tag{A9}$$

In these equations, $R_{\max,x}^i$ and $R_{\max,v}^i$ are given by equations (A1) for each cloud belonging to the cell, $(x_{\mathrm{G}}^i, v_{\mathrm{G}}^i) \in [(x_1, v_1), (x_2, v_2)]$.

To construct the list of clouds contributing to point (x, v), one starts from the root and walks down into the tree. During the walk, only cells satisfying $(x, v) \in C_{\rm eff}$ are opened, i.e. decomposed in four subcells, until convergence is achieved, i.e. when the cell contains zero or one cloud. This algorithm is quite fast: the loss of speed compared to the trivial method explained in Section A1 is of the order of $\ln\ N_{\rm tot}$, where $N_{\rm tot}$ is the total number of clouds. It has the advantage of being very general since it allows us to compute $f(x, v)$ at any point. This explains why we adopted it from the very beginning when we started to develop our tree-code, in order to keep this latter as flexible as possible.

APPENDIX B: REFINEMENT PROCEDURE: TECHNICAL DETAILS

This appendix deals with technical issues of our refinement procedure. In Section B1, we explain in detail how the refinement structure is implemented. In particular, as in standard ART methods, to keep the method stable, we make sure that transitions between refinement levels are not too abrupt. We also take into account the large extension of the clouds. In Section B2, we describe the way we set our criteria of refinement, namely one based on local curvature and the other one based on local convergence.

B1 Principle of refinement

Let L be the level of refinement, equal to $L = 0$ for the coarse level. For simplicity, we assume that the coarse level is a grid with fixed interspacing, $\Delta_x = \Delta_v = \Delta_{\rm g}$. In order to avoid instabilities, the function $L(x, v)$ should vary smoothly in phase space: abrupt transitions between two levels $|L_1 - L_2| > 1$ are forbidden. Also, special care has to be taken of the large extension of the clouds. For example, assume that we split a cloud of size $R = \sqrt{2}/2\Delta_{\rm g}$ with cut-off radius $R_{\rm max} = 4R$ into four twice smaller clouds. Since the cloud influences remote regions beyond the cut-off scale of the four subclouds, refinement should be performed as well in these regions. Fig. B1 shows how to handle this problem and naturally to preserve smoothness of the function $L(x, v)$. While following the rules dictated by the procedure detailed in Fig. B1, we use a criterion based on local convergence or/and on measurement of local curvature to decide if a cloud has to be refined or not, as explained later in Section B2.

Once a full hierarchy of clouds and subclouds has been created, we proceed as follows to reconstruct the phase-space distribution function.

(o) Set up the coarse level, $L = 0$: compute f_i^0 from the old set of clouds (prior to remap) at sampling points $(x_{\rm G}^{i,0}, v_{\rm G}^{i,0})$. Then use the van Citter algorithm to compute the masses, M_i^0, of the coarse-level clouds, as explained in Section 3.1.2;

(i) If the maximum level of refinement is not reached, increase L by one, otherwise stop.

(ii) Compute f_i^L at the positions of clouds of level L, $(x_{\rm G}^{i,L}, v_{\rm G}^{i,L})$, from the old set of clouds. Now, consider clouds of level L like residues and use the van Citter algorithm to compute the masses of these clouds, M_i^L. To estimate $\tilde{f}_i^L$ in equation (63), clouds of lower levels $\ell \leqslant L - 1$ contribute, but with fixed masses computed previously. Start again with step (i).

In step (ii), once the cloud masses have been determined up to some level, they remain unchanged while determining the masses of clouds of upper levels. It is indeed expected that the corrections brought by residues improve the reconstruction of the distribution function within the corresponding refinement region, without contaminating the remaining part of phase space.

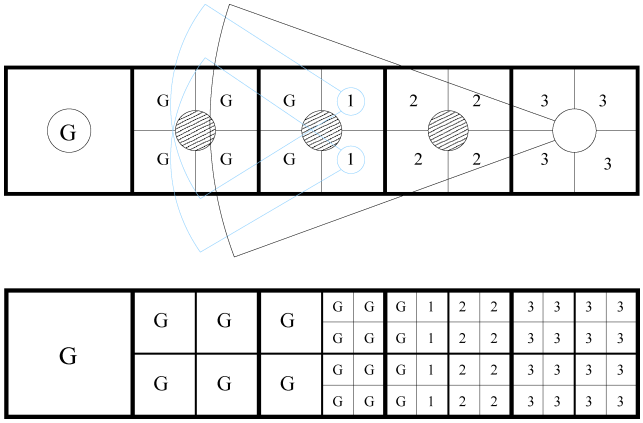

Figure B1. Sketch of our refinement procedure, for our choice of cloud size, R_L, compared to local cloud interspacing, $\Delta_{\rm g}/2^L$: $2^L R_L/\Delta_{\rm g} = \sqrt{2}/2$, which implies a $4R_L$ cut-off at $2^L R_{\rm max}/\Delta_{\rm g} = 2\sqrt{2}$. Suppose we decide to refine the right-most cloud of the upper panel, of level $L - 1$. This cloud is supplemented with four twice smaller subclouds of level L, labelled by 3. This, however, influences a region of radius $R_{\rm max}/2^{L-1}$, represented by the black circular arc. To enforce our refinement criterion on all the region influenced by this cloud, we must add supplementary layers of subclouds around it, labelled by 2 and by 1. We can stop adding corrections at subclouds labelled by 1, since their extension covers the zone of influence (light circular arcs). For easy implementation of regularization (B2), we also create ghost clouds of zero mass designated by the letter 'G' in order to compute the correct weights for the refined level. With this refinement procedure, we see that the cloud to be refined has to be sufficiently far away from ghosts. The closest possible ghost for the right-most cloud at level $L - 1$ is the left-most one, which was labelled by a G surrounded by a circle. Our refinement can be performed recursively to higher level, as illustrated in the lower panel, by obeying the following simple rules: (1) it is forbidden to refine ghosts; (2) clouds labelled by 1 can only be refined in ghosts; (3) clouds labelled by 2 can only be either refined in ghosts, in subclouds labelled by 1 or in subclouds labelled by 2 according to their distance to the closest ghosts, as illustrated in the upper and lower panel; (4) clouds labelled by 3 can be fully refined. Note that, by construction, the region labelled with 3's can be refined to arbitrary level L_∞, while keeping smooth transitions between coarse level 0 and L_∞.

In our implementation of refinement, we use the van Citter deconvolution algorithm. Alternatively, one might use the Lucy method. In that case, clouds with $L \geqslant 1$ can no longer be considered as residues, and, due to the multiplicative nature of the Lucy algorithm, we would have to reiterate at all levels to compute the overall distribution of masses, M_i^ℓ, $\ell \leqslant L$. This would complicate the algorithm considerably, with no guarantee of easy convergence, so we decided for this work not to test this refinement scheme.

To compute $\tilde{f}_i$ in equations (63) and (64) from the new set of round clouds, one can still use equation (59), as discussed at the end of Section 3.1.3. It now simply reads

$$\tilde{f}(x, v) = \sum_{\ell=0}^{L} \frac{1}{V^\ell} \sum_i M_i^\ell\, G_\ell\left(x - x_{\rm G}^{i,\ell}, v - v_{\rm G}^{i,\ell}\right), \qquad \text{(B1)}$$

where $V^\ell = V^0\, 4^{-\ell} = \Delta_x \Delta_v 4^{-\ell}$ and the function G_ℓ is the same as the function G but with a radius $R_\ell = 2^{-\ell} R$. To estimate f_i^L from the ancient set of clouds, one can similarly adapt equation (66), which

now becomes

$$f(x,v)=\sum_{\ell=0}^{L}\frac{W}{V^{\ell}}\frac{\sum_i G_\ell\left(x-x_{\rm G}^{i,\ell},v-v_{\rm G}^{i,\ell}\right)M_i^{\ell}}{\sum_i G_\ell\left(x-x_{\rm G}^{i,\ell},v-v_{\rm G}^{i,\ell}\right)}. \tag{B2}$$

We see in equation (B2) that the interpolation is performed at each level of refinement separately. In order to have proper normalization, we need to add at the border of each refinement level a layer of ghost clouds with zero masses but contributing to the weight in the denominator, as described in Fig. B1.

B2 Criteria of refinement

Our refinement is by essence set up to follow details of the distribution function where needed. Consider the distribution function as a surface of equation $z=f(x,v)$. The local curvature of this surface determines to what extent these details can be reproduced by our clouds of finite size. If the size R of the clouds is not small enough compared to the local minimum curvature radius $\mathcal{R}_{\rm c}$, we expect significant loss of details as well as aliasing effects. A first natural refinement criterion thus relies on the value of $R/\mathcal{R}_{\rm c}$. However, it is expected, equivalently, that deconvolution will have trouble converging when $R/\mathcal{R}_{\rm c}\gtrsim 1$, since details cannot be adjusted correctly by the clouds. This motivates an alternative criterion of refinement, based on local convergence of the reconstruction. The convergence criterion also has the advantage of being a quite natural extension of our approach: it guarantees by definition the quality of the reconstruction, at variance with the local curvature criterion. This latter indeed relies on the measurement of second derivatives of f, which can be quite noisy. However, as illustrated by Section 6.2, both refinement criteria give very similar results.

While constructing the hierarchy of clouds and subclouds, we proceed level by level. To create clouds of level L, clouds of level $L-1$ are tested as follows.

(A) *Local convergence criterion*. In this case, it is necessary to reconstruct the distribution function simultaneously with the refinement structure, using steps (o), (i) and (ii) described in Section B1. Put simply, step (i) has to supplemented with the criterion on local convergence to create the new level of refinement, L: one sets a list of clouds of level $L-1$ for which the reconstruction scheme did not converge, $\delta_i^{L-1}>\delta_{\rm c}^{L-1}$ in equation (63), after some fixed number of iterations, say 10 according to the discussion in Section 3.1.2 In principle, the optimal calculation of $\delta_{\rm c}^{L-1}$ as a function of L depends on the noise properties of the reconstruction, but it would go beyond the scope of this paper to analyse them in detail. We therefore consider a very simple way of setting the convergence threshold:

$$\delta_{\rm c}^{L}=\delta_{\rm c}^{0},\quad L\geqslant 1, \tag{B3}$$

which ignores possible propagation of errors in the reconstructions from level to level, at the risk of accumulating small-scale artefacts. For this reason, the value of $\delta_{\rm c}^0$ should in practice be slightly larger than discussed in Section 3.1.2, to avoid unnecessary refinement due to fluctuations of the small-scale noise. A good practical value is $\delta_{\rm c}^0\simeq 0.001$.

(B) *Local curvature criterion*. The clouds of level $L-1$ for which the local curvature of the phase-space distribution function is larger than some threshold are refined. The local curvature is estimated at cloud centres from the old set of clouds (prior to remap), using the Hessian of the function $f(x,v)$ given by equation (B2) (we do not write it here for simplicity). Our refinement criterion is

$$\left(\frac{\Delta_{\rm g}}{2^{L-1}}\right)^2\max(|\lambda_1|,|\lambda_2|)>F_{\rm curvature}, \tag{B4}$$

where λ_1 and λ_2 are the eigenvalues of the Hessian. A good practical choice of the refinement parameter is $F_{\rm curvature}\simeq 0.1$.

(C) *Keep stability of the refinement*. At positions of clouds of level $L-1$ to be refined, check locally what the refinement level $L_{\rm old}$ was in the old set of clouds and impose the condition that $L\leqslant L_{\rm old}+1$. To do that, we consider a square of size $\Delta_{\rm g}/2^{L-1}$ centred on $(x_{\rm G}^{i,L-1},v_{\rm G}^{i,L-1})$ and find with standard quad-tree search the set of old clouds intersecting with it. $L_{\rm old}$ corresponds to the maximum refinement level found in this set. For stability purposes, we should have $L\leqslant L_{\rm old}+1$. During run-time, we indeed do not expect the creation of more than one level of refinement per remap. If that happens, it must be due to some spurious small-scale artefact created at the moment of the previous remap, which we do not want to propagate furthermore.

There is a subtlety in our algorithm that we have to mention now. Indeed, the criteria explained above are applied to a discrete set of sampling points. To make sure that this sampling catches all the features of the distribution function, one has to examine locally conditions (A) and (B) at *all* levels up to $L_{\rm old}$, by creating the corresponding subclouds. Due to the large extension of the clouds and the way $L_{\rm old}$ is computed, a large number of subclouds is in fact unnecessary. At the end, the hierarchy of clouds has to be 'cleaned up', to keep only the relevant refinement regions, i.e. regions satisfying criteria (A) or/and (B). After being cleaned up, the new set of clouds and subclouds is ready for successive deconvolutions at various levels, which are thus performed twice when the convergence criterion is used: once to construct the hierarchy of clouds and once again after this hierarchy has been cleared of unnecessary subclouds.

APPENDIX C: EXPLICIT TWO-DIMENSIONAL CLOUD EQUATIONS

$$\frac{{\rm d}\lambda_{11}}{{\rm d}t}-4\,\phi_{11}\lambda_{31}-4\,\phi_{21}\lambda_{41}=0, \tag{C1}$$

$$\frac{{\rm d}\lambda_{22}}{{\rm d}t}-4\,\phi_{12}\lambda_{32}-4\,\phi_{22}\lambda_{42}=0, \tag{C2}$$

$$\frac{{\rm d}\lambda_{33}}{{\rm d}t}+2\,\lambda_{13}=0, \tag{C3}$$

$$\frac{{\rm d}\lambda_{44}}{{\rm d}t}+2\,\lambda_{24}=0, \tag{C4}$$

$$\frac{{\rm d}\lambda_{21}}{{\rm d}t}-2\,\phi_{12}\lambda_{31}-2\,\phi_{22}\lambda_{41}-2\,\phi_{11}\lambda_{32}-2\,\phi_{21}\lambda_{42}=0, \tag{C5}$$

$$\frac{{\rm d}\lambda_{31}}{{\rm d}t}+\lambda_{11}-2\,\phi_{11}\lambda_{33}-2\,\phi_{21}\lambda_{43}=0, \tag{C6}$$

$$\frac{{\rm d}\lambda_{32}}{{\rm d}t}+\lambda_{12}-2\,\phi_{12}\lambda_{33}-2\,\phi_{22}\lambda_{43}=0, \tag{C7}$$

$$\frac{{\rm d}\lambda_{41}}{{\rm d}t}+\lambda_{21}-2\,\phi_{11}\lambda_{34}-2\,\phi_{21}\lambda_{44}=0, \tag{C8}$$

$$\frac{{\rm d}\lambda_{42}}{{\rm d}t}+\lambda_{22}-2\,\phi_{12}\lambda_{34}-2\,\phi_{22}\lambda_{44}=0, \tag{C9}$$

$$\frac{{\rm d}\lambda_{43}}{{\rm d}t}+\lambda_{23}+\lambda_{14}=0. \tag{C10}$$

This paper has been typeset from a TEX/LATEX file prepared by the author.