VINE – A numerical code for simulating astrophysical systems using particles I: Description of the physics and the numerical methods

M. Wetzstein[1,2], Andrew F. Nelson[3,4], T. Naab[2,5] and A. Burkert[2]
mwetz@usm.lmu.de

ABSTRACT

We present a numerical code for simulating the evolution of astrophysical systems using particles to represent the underlying fluid flow. The code is written in Fortran 95 and is designed to be extremely versatile, flexible and extensible, with modular options that can be selected either at the time the code is compiled or at run time through a text input file. We include a number of general purpose modules describing a variety of physical processes commonly required in the astrophysical community and we expect that the effort required to integrate additional or alternate modules into the code will small. In its simplest form the code can evolve the dynamical trajectories of a set of particles in two or three dimensions using a module which implements either a Leapfrog or Runge-Kutta-Fehlberg integrator, selected by the user at compile time. The user may choose to allow the integrator to evolve the system using individual timesteps for each particle or with a single, global time step for all. Particles may interact gravitationally as N-body particles, and all or any subset may also interact hydrodynamically, using the Smoothed Particle Hydrodynamic (SPH) method by selecting the SPH module. A third particle species can be included with a module to model massive point particles which may accrete nearby SPH or N-body particles. Such particles may be used to model, e.g., stars in a molecular cloud. Free boundary conditions are implemented by default, and a module may be selected to include periodic boundary conditions. Cosmological expansion may be included by selecting another module. We use a binary 'Press' tree to organize particles for rapid access in gravity and SPH calculations. Modules implementing an interface with special purpose 'GRAPE' hardware may also be selected to accelerate the gravity calculations. If available, forces obtained from the GRAPE coprocessors may be transparently substituted for those obtained from the tree, or both tree and GRAPE may be used as a combination GRAPE/tree code. The code may be run without modification on single processors or in parallel using OpenMP compiler directives on large scale, shared memory parallel machines. In comparison to the Gadget-2 code of Springel (2005), the gravitational force calculation, which is the most costly part of any simulation including self-gravity, is $\approx 3.5 - 4.8$ times faster with VINE when run on 8 Itanium 2 processors in an SGI Altix, while producing nearly identical outcomes in our test problems. We present simulations of several test problems, including a merger simulation of two elliptical galaxies with 800000 particles. The code is available to the public under the terms of the Gnu General Public License.

Subject headings: methods: numerical — methods: N-body simulations — galaxies: interactions

[1]Department of Astrophysical Sciences, Princeton University, Princeton, NJ 08544, USA

[2]Universitäts-Sternwarte, Scheinerstr. 1, 81679 München, Germany

[3]Los Alamos National Laboratory, HPC-5 MS B272, Los Alamos NM, 87545, USA

[4]UKAFF Fellow

[5]Institute of Astronomy, Maddingley Road, Cambridge, United Kingdom

1. Introduction

In modern astrophysics, the numerical simulation of systems whose complexity is beyond the capabilities of analytical models has become a widely used tool. On nearly all length scales, ranging from problems on cosmological distances, to galaxy formation and evolution to star and planet formation, numerical simulations have contributed much to our current understanding of the physical processes which lead to the universe we observe.

The numerical simulation of a self-gravitating and/or gas dynamical system is a basis common to all those problems, no matter what length scale they belong to. The simulation techniques for such systems can be divided into two different approaches: grid based methods divide space into finite sized cells and compute the physical quantities such as temperature, pressure, etc., inside those cells (see e.g. Stone & Norman 1992; Ryu et al. 1993; O'Shea et al. 2004, and references therein). Particle based methods represent a system by a set of particles to which physical quantities such as mass, position and velocity are assigned or computed (see e.g. Hernquist & Katz 1989; Dave et al. 1997; Springel et al. 2001; Wadsley et al. 2004; Springel 2005, and references therein). Which approach is best for modeling a particular system depends both on the problem to be modeled and the biases of the researcher doing the modeling. Without going into the details of relative merits and shortcomings of either approach, we point out that for some problems, a grid based approach may be nearly unfeasible because of the existence of irregular boundaries. Large voids can also be problematic for a grid based simulation because it requires a large number of empty or nearly empty and uninteresting zones be included at a high computational expense. A particle based simulation naturally concentrates the computational work in the most interesting areas of the flow, in most cases a very valuable feature, but the absence of particles in voids can be problematic if the simulation requires such low density regions to be resolved at high accuracy (see e.g. O'Shea et al. 2005, for a recent comparison of grid and particle based codes for cosmological simulations). It may also suffer from a relatively poorer reproduction of the fluid behavior at shocks.

For a system evolving only under the influence of gravity, a particle based approach leads to the classical N-body problem. A set of N_p particles evolve according to the force on each particle exerted by all the others. The efficient computation of these forces is a longstanding numerical problem. If gas dynamics is also required, the Smoothed Particle Hydrodynamic (SPH, Lucy 1977; Gingold & Monaghan 1977; Benz 1990; Monaghan 1992) method has achieved great success at incorporating such processes into the framework of a particle method. In SPH, the gas is also represented using particles (which makes the method so useful in combination with N-body methods), which are assumed to sample the local hydrodynamic quantities of the underlying flow. In addition to a position and velocity, these particles also possess an internal energy intrinsic to each and a volume (or surface) density that is reconstructed for each particle, based on the positions of nearby 'neighbor' particles. Thus, these gas particles feel not only the gravitational forces that all particles in the simulation do, but also pressure forces and other gas dynamical effects.

Modeling dynamical and hydrodynamical systems using particles relies on the sufficiently accurate computation of both the gravitational and hydrodynamical forces of the particles on each other, then advancing them forward in time according to those forces. Thus a time integration of the particles' equations of motion and additional equations for hydrodynamic quantities such as internal energy, is the problem to be solved. Constraints on the time integration are that we would like it to faithfully reproduce the evolution of the real system that the simulation is supposed to model, and that it do so efficiently, so that results may be obtained quickly and insight into the physical world gained at a minimum of cost.

In this paper and a companion (Nelson, Wetzstein, & Naab 2006, hereafter Vine2), we describe a numerical code for efficiently simulating the evolution of astrophysical systems using N-body particles, with the optional additions of including gas dynamical effects using the SPH method, self gravity and additional massive 'star' particles which may accrete the other species of particles. We call this code VINE. The present paper describes the physics we have implemented, the high level code design and the results of sim-

ulations using the code on a number of test problems as well as a comparison to the Gadget-2 code of Springel (2005). Vine2 describes the low level design and optimization of the most computationally expensive parts of the code, the methods used to parallelize it and the performance of each part in serial and in parallel. VINE has been succesfully used for a large series of simulations on galaxy interactions (Naab & Burkert 2003; Jesseit et al. 2005; Burkert & Naab 2005; Dasyra et al. 2006a,b; Naab et al. 2006a,b; Naab & Trujillo 2006; Bell et al. 2006; Thomas et al. 2007; Burkert et al. 2007; Naab et al. 2007; Jesseit et al. 2007; Wetzstein et al. 2007) as well as planet formation (e.g. Nelson 2006).

In section 2 we describe the implementation of two second order integration methods included with the code, and a discussion of the criteria used to determine the time steps used to evolve the particles forward in time. We describe in sections 3 and 4, respectively, the form of the equations used to implement the SPH method and the different options for the calculation of gravitational forces. In section 5, we describe the implementation of 'star' particles which can accrete the N-body and SPH particle species. Modifications of the equations of motion required to model cosmological expansion are given in section 6 and in section 7 we describe the boundary conditions available in the code. In section 8 we demonstrate some of the capabilities of the code on several test simulations, including both SPH as well as a pure N-body problems. In section 9 the performance of VINE is compared to that of the Gadegt-2 code of Springel (2005). Finally, in section 10, we summarize the features of our code and give web sites where the source code may be obtained electronically.

2. Time integration

In order to simulate the evolution of a physical system using a set of N_p particles, we require first a set of equations by which the system evolves and second a method for integrating the system forward in time. In the case of particle systems involving only gravitational interactions ('N-body' simulations), the equations form a set of coupled first order differential equations governing the motion of those particles in response to each other and (if present) to outside influences. This set consists of the equations describing the motion of each particle,

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i \tag{1}$$

and the equations for momentum conservation

$$\frac{d\mathbf{v}_i}{dt} = -\frac{\nabla\Phi}{m_i}. \tag{2}$$

The solution of these equations is difficult because each of the $4N_p$ or $6N_p$ equations (in 2 or 3 dimensions) is coupled through the gravitational potential Φ. Other coupling terms must be added in cases where other physical phenomena such as hydrodynamics (section 3) or cosmology (section 6) are active, and may require additional equations be solved. Hydrodynamic systems, for example, must solve not only a momentum equation appropriately generalized from above, but also mass and energy conservation equations, and cosmological evolution will require other modifications to the equations to account for cosmological expansion.

A variety of methods for integrating differential equations are available (see e.g. the textbooks of Hockney & Eastwood 1981; Fletcher 1997), with varying degrees of utility for any given problem. Determining which method is most efficient is highly dependent on the characteristics of the problem itself. Most astrophysical systems, for example, develop highly non-linear flow patterns with the practical consequence that high order integration methods (i.e. those with a mathematical truncation error proportional to a high power of the integration step size) are not generally useful. The nonlinearities mean that time steps must be restricted to very small sizes in order to resolve the flow, while the high order integration requires many derivative calculations per timestep, yielding a very high computational cost to evolve a system for a given amount of time. Even among integrators of identical 'order', characteristics of the errors that develop can vary, with one being inappropriate for use on a problem another might be ideally suited to solve.

In order to allow the user enough flexibility to determine what is best for a specific problem, we have implemented both a second order Runge-Kutta scheme (Fehlberg 1968) and a second order leapfrog scheme (see e.g. Hockney & Hohl 1969; Hernquist & Katz 1989; Rasio & Shapiro

1991; Springel et al. 2001) in VINE. Although very different in structure (leapfrog requires only one force computation per time step for example, while the Runge-Kutta scheme requires two), users may transparently select one or the other at the time the code is compiled. Both integrators are very modular in the sense that they use the same bookkeeping scheme for particles and their timesteps, and identical calls to update routines. If a user finds that still a different choice of integrator is required, we expect that it would be straightforward to add it as an alternative as well.

2.1. The Runge-Kutta-Fehlberg (RKF) Integrator

Runge-Kutta schemes of a variety of forms have been developed since the original publication of the general method more than 100 years ago (Kutta 1901), but until the work of Fehlberg (1968) they included no formal description of the size of errors that developed during an integration. Fehlberg realized that a Runge-Kutta scheme of a given truncation order could be embedded in a similar scheme of one order higher, given a suitable choice of coefficients for both. The resulting pair of methods, used together, could be used to determine a limit on the size of the next order truncation error for the lower order scheme in the pair. In VINE, we have implemented the first order scheme with second order error control ('RKF1(2)'), as described by Fehlberg (1968). A brief description of the scheme is summarized here.

For any quantity q to be integrated in time, the quantity q^{n+1} at the new time t^{n+1} is computed from its value q^n at the previous time t^n utilizing the discretization:

$$q^{n+1} = q^n + (c_0 k_0 + c_1 k_1)\Delta t^n \tag{3}$$

where the c_i are constant parameters and the k_i are time derivatives of q evaluated at various points during the timestep:

$$k_0 = \dot{q}(t^n, q^n), \tag{4}$$

$$k_1 = \dot{q}(t^n + \alpha_1 \Delta t^n, q^n + \beta_{10} k_0 \Delta t^n) \tag{5}$$

$$k_2 = \dot{q}(t^n + \alpha_2 \Delta t^n, q^n + \beta_{20} k_0 \Delta t^n + \beta_{21} k_1 \Delta t^n). \tag{6}$$

where $\dot{q}$ is the time derivative of q and Δt^n is the time step from t^n to t^{n+1}. The k_2 term does not appear directly in the integration equation 3 above, but does appear in the error criterion defined in section 2.3.2 below. The coefficients α_k and β_{kl} and c_k defined by Fehlberg are reproduced in table 2.1. By definition of the coefficients, the k_2 term is identical to the k_0 term for the following timestep, reducing the number of derivative evaluations to two per timestep. The c_k define the coefficients used by the first order RK scheme, while the $\hat{c}_k$ terms define the coefficients used in the second order scheme used only indirectly to define the truncation error.

2.2. The Leapfrog Integrator

The leapfrog (LF) integration scheme is formally an offset integrator: positions and velocities are offset from each other in time by half a time step (see e.g. Hockney & Eastwood 1981). Alternate updates of position and velocity advance from one half step behind to one half step ahead of the other update in the sequence, effectively 'leapfrogging' over each other in the integration scheme, which takes its name by analogy from the children's game. The leapfrog implementation in VINE is similar to that of Springel et al. (2001), for which a mathematically equivalent form is used, in which the equations for the positions and velocities are written in a non-offset form as

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \mathbf{a}^{n+1/2}\Delta t^n \tag{7}$$

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \frac{1}{2}\left(\mathbf{v}^n + \mathbf{v}^{n+1}\right)\Delta t^n \tag{8}$$

where again indices n, $n+1/2$ and $n+1$ denote quantities at time t^n, $t^{n+1/2}$ and t^{n+1}, respectively, and Δt^n is the step from n to $n+1$. To recover the offset form, notice that positions and accelerations are actually defined on half timesteps, but that the position update is effectively split into two halves. With a fixed increment Δt, each position update as defined in equation 8 uses the velocity corresponding to two separate velocity updates, half from timestep n, $\mathbf{v}^n/2$, and half from

Table 1: Coefficients for the RKF1(2) Integrator

k	α_k	β_{k0}	β_{k1}	c_k	$\hat{c}_k$
0	0			1/256	1/512
1	1/2	1/2		255/256	255/256
2	1	1/256	255/256		1/512

timestep $n+1$, $\mathbf{v}^{n+1}/2$, so that, effectively, updates of position are only half completed at any 'full' time step n.

The velocity update requires that accelerations be calculated on half steps, $n+1/2$. For simulations involving self gravity and hydrodynamics, the accelerations depend on particle positions, so that a separate, temporary update of the position to its correctly offset temporal location is required. This update takes the form

$$\mathbf{x}^{n+1/2} = \mathbf{x}^n + \frac{1}{2}\mathbf{v}^n \Delta t^n \tag{9}$$

as expected from the discussion above. Other quantities requiring integration, such as internal energy, smoothing lengths or viscous coefficients needed for hydrodynamic simulations (section 3), are defined on integer time steps. Their derivatives must therefore be calculated on half time steps at the same time as the accelerations themselves are calculated. Complications arise because for most such variables, the derivative is a function of the variable itself or of others defined on integer time steps. Two simple examples are of artificial viscosity or of PdV work, each of which require velocity. VINE employs a linear extrapolation of each quantity from n to $n+1/2$, as shown in equation 9 for position, so that the integration scheme itself remains formally second order. In summary, the algorithm can be written as

1. complete position update to $\mathbf{x}^{n+1/2}$, extrapolate other quantities as required.
2. compute $\mathbf{a}^{n+1/2}$ and other derivatives.
3. update velocities $\mathbf{v}^n \rightarrow \mathbf{v}^{n+1}$, using equation 7. Update other relevant quantities using appropriate analogous update equations.
4. update positions $\mathbf{x}^n \rightarrow \mathbf{x}^{n+1}$, using equation 8.

After the fourth step the sequence starts anew.

Although slightly more cumbersome than the offset form, the leapfrog variant above is advantageous because adjustable time steps, such that $\Delta t_n \neq \Delta t_{n+1}$, are straightforward to implement in the form above, as are individual time steps for different particles (see section 2.4). Both features will be desirable in simulations of systems where time scales vary widely as conditions change over time. The consequences for such adaptability is that the exact leapfrog symmetry between position and velocity updates is lost, but changes between one time step value and another should be infrequent enough in practice to make overall errors resulting from them small.

2.3. Timestep Criteria

In order to produce an accurate integration, time steps must be chosen that are small enough to maintain the stability of the system against the growth of errors. At the same time, time steps should not be much smaller than required to maintain stability and accuracy, because it wastes computational resources that could be more efficiently employed in performing larger simulations. Here we describe the criteria used in VINE to determine timesteps for the particles.

2.3.1. Time Step Criteria common to both the Leapfrog and RKF Integrators

The timestep criteria described in this section apply to both the leapfrog and the RKF integrators. The next time step Δt^{n+1} of a particle i is determined by the minimum of value derived from a set of N criteria:

$$\Delta t^{n+1} = \min_N(\Delta t_N^{n+1}), \tag{10}$$

where we have suppressed the subscript, i, designating each particle. Whether or not to include a particular criterion may be selected by the user at compile time by commenting out (or not) calls to subroutines that calculate one or another of the Δt_N^{n+1}, and by routines active only when certain options are selected, such as the Runge-Kutta integrator or SPH (sections 2.3.2 and 3.4).

Three simple criteria are based on changes in the acceleration of a particle:

$$\Delta t_a^{n+1} = \tau_{acc}\sqrt{\frac{h}{|\mathbf{a}|}}, \tag{11}$$

its velocity:

$$\Delta t_v^{n+1} = \tau_{vel}\frac{h}{|\mathbf{v}|}, \tag{12}$$

or both in combination:

$$\Delta t_{va}^{n+1} = \tau_{va}\frac{|\mathbf{v}|}{|\mathbf{a}|}, \tag{13}$$

where h, $\mathbf{a}$ and $\mathbf{v}$ are the gravitational softening length, the acceleration and velocity of the particle i at the previous time step, respectively, and the three values of τ are tuning parameters for each criterion. Numerical experiments show that $\tau_{acc} \approx 0.5$ gives good results. When included, we use similar values for the other two tolerance parameters as well.

Although the combination of all three criteria is sometimes useful, and indeed is used in e.g. Nelson (2006) with VINE, in many cases it is sufficient to include only the acceleration based criterion of equation 11, allowing the others to be neglected. For example, when the velocity criteria are included they can impose very restrictive constraints on the calculated time step. If a particle moves at very high velocities, equation 12 can require small time steps even when the particle does not change its trajectory and could otherwise be integrated with large time steps. Similarly, equation 13 can limit the time step of a particle when it moves very slowly but feels only small forces.

2.3.2. Time Step Criteria for the RKF Integrator

In addition to the conditions above, the RKF integrator requires an additional criterion, which limits the second order truncation error in the discretization (see section 2.1). As defined by Fehlberg, the second order truncation error for integrating variable q through a time $\Delta t^{(i)}$ will be:

$$TE = \hat{c}_2(k_0 - k_2)\Delta t^n, \tag{14}$$

where k_0 and k_2 are defined in equations 4 and 6 and $\hat{c}_2$ is defined in table 2.1. Unfortunately, the truncation error as defined is an absolute error. It depends on the units for a given variable as well as the size of the system and is therefore not particularly useful without explicit tuning for every variable, physical system and simulation. Various relative error metrics are straightforward to develop from equation 14 however. For example, we may define a relative error metric such that the truncation error is no larger than a small fraction of the magnitude of the variable itself:

$$RE = \frac{|TE|}{\tau_{\mathrm{RKF}}|q|}, \tag{15}$$

where we define τ_{RKF} as a tunable parameter restricting the error, and we require RE to be $\lesssim 1$ for an acceptable error.

We may expect the optimal timestep to be proportional to the square root of the truncation error, since the error itself is second order in time. Then we may determine a new time step from the old by comparing the ratio of the new and old values to the error metric:

$$\Delta t^{n+1}_{\mathrm{RKF}} = \Delta t^{n}_{\mathrm{RKF}} \sqrt{1/RE}. \tag{16}$$

With this definition, the $n+1$ time step will be decreased when the error for a given time step is large. If small, it will be increased. We define the final value of $\Delta t^{n+1}_{\mathrm{RKF}}$ to be the minimum over all integration variables defined in the simulation.

Although often an improvement over the direct measure of error and time step definition, equation 16 may still suffer from several deficiencies in practice, depending on the specific integration variable. For example, the size of the timestep calculated for positions depends on the position itself, and particles near the origin will necessarily receive more restrictive time steps than those further away. An arbitrary change of coordinate system, shifting the entire system some distance in any direction, will also change the error metric and time step calculation. For the same reasons, velocity errors and timesteps will suffer similar problems.

A variety of strategies to sidestep undesirable properties for one variable or another are available, including replacing q in equation 15 with its value subtracted from the system's average velocity or center of mass for velocity or position coordinates respectively, or adding a constant error softening value to eliminate singularities in the error near zeros of the variable. Following discussion in Press et al. (1992), one may also replace $|q|$ for some variables with its value added to its change at the last timestep:

$$q' = |q| + |\Delta t^n \dot{q}|. \tag{17}$$

Alternate error metrics such as these have been implemented in VINE with some success on several systems we have studied, however in general, we expect that suitable error metrics will need to be worked out on a case by case basis by the user. Some small comfort may be had in the fact that, under most conditions, other error conditions are more restrictive than the RKF error, making questions of the suitability of the form of the RKF criterion moot.

2.3.3. Differences in settings for time step criteria when global or individual time steps are used, or for different problems

For both of VINE's integrators, it is possible to check after each time step whether the integration over that time step met or failed the set of error criteria described above. If so, in principle one can revert the time step and repeat it with a smaller step size. In practice, reversion is only possible if the entire system is advanced using a single, global time step for all particles, and is in fact done in VINE when the global time step option is selected. When individual time steps are used (discussed in section 2.4 below), reverting the time step is usually not possible because it requires keeping track of a large set of previous time steps for every particle in the system, and is therefore usually prohibitive in terms of memory as well as computational effort. Thus for an individual time step scheme, the criteria for choosing the next time step and their settings must be chosen more conservatively than with the global time step option, in order to ensure in advance that the time step is small enough to integrate that particle's properties correctly. For example, Bate et al. (1995) demonstrated that criteria similar to those in equation 16 give good results with the RKF integrator when used alone with global timesteps, but that errors become unacceptable when used alone with individual particle time steps. Adding another criterion of the form of equation 11 alleviated the problem. Similar situations may arise with the criteria currently implemented in VINE when used on different problems. We have therefore designed the error criteria code as a set of independent routines for calculating specific error criteria, each with the same interface and each called from a master routine, whose sole purpose is to serve as a location at which criteria may be included or excluded. The selection of which criteria to use and the addition of other criteria can be done by the user with minimal difficulty, and the timestep itself is computed from the minimum of all active criteria.

2.4. The Individual Timestep Scheme

In many astrophysical contexts, it is necessary to model the evolution of regions with densities (or other quantities) that are orders of magnitude larger than those of other regions in the same simulation, or that change orders of magnitude more quickly. Such large variations naturally introduce a wide range of physical timescales.

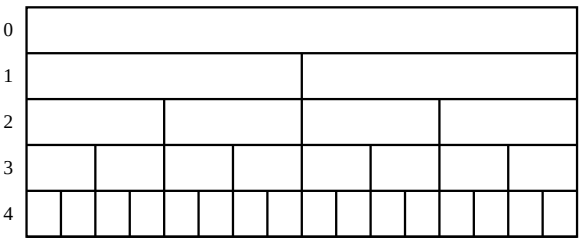

Fig. 1.— The individual time step scheme used in VINE. Only the five highest levels of the time step hierarchy are shown here. Level 0 represents the maximum time step, level 4 is a factor of 2^4 smaller.

Although it might be desirable in some cases to evolve all particles with a single time step in order to maintain a highly stable integration, the computational expense of doing so in all cases can be prohibitive. Instead, it is possible to assign time steps for each particle on an individual basis as required by a given particle and thereby to evolve the particles independently. Assigning individual time steps to the particles can speed up a simulation considerably, since little processor time is wasted on evolving less dynamic regions with the same small time steps as the most dynamic regions present in the system.

Users of VINE can select whether to run simulations with either a single global timestep or an individual time step for each particle. Global time steps can be set either to the minimum absolute time step ('global continuous' mode) or to the minimum binned time step ('global binned' mode) corresponding to the time bins that would be used if the individual time step option were active, as described below. The latter option restricts the time steps to a discrete set of sizes and therefore may enhance integrator stability, particularly for the leapfrog integrator, in which fixed time steps are formally required to retain the symplectic character of the integration.

When the user selects individual time steps, VINE uses a variant of the scheme proposed by Hernquist & Katz (1989) for their TreeSPH code, based on previous work of Porter (1985) and Ewell (1988). The user first chooses a maximum time

step allowed for the system, which forms the top of the hierarchy of smaller time step levels constructed by dividing the time step on the next higher level by 2 (see figure 1). For time increments smaller than the maximum, time is regarded as an integer quantity, whose maximum value is represented by a large power of 2, in VINE set to 2^{28}. Each shorter time step can then be represented by a value 2^{28-n} where n is the time step level counted from the top (see figure 1). Multiplying each integer by the smallest real valued time step increment recovers the true, real valued time or time increment, Δt, as needed. Time updates smaller than the maximum are updated in integer increments, thereby avoiding errors associated with time drift or finite precision truncation errors, as may occur when many small, real valued quantities are added together. After the integration has proceeded through a full step, so that particles on all levels have been updated to the end of the largest step, the real valued absolute time is incremented by the value of the largest step.

VINE assigns each particle its own provisional time step as described in 2.3, then truncates it to the next smaller time step level defined in the scheme. Time step assignment is done transparently for both the leapfrog and RKF integrators using the same binning scheme. Because the number of levels is finite, particles evolve forward in groups corresponding to one or more levels of the scheme that are currently active, rather than one by one according to a continuous spectrum of time steps. This is important for achieving computational efficiency, both in serial and parallel operation, because force computations for only one or a few particles at a time are comparatively inefficient and difficult to balance among a large array of processors running in parallel. Overheads associated with repeatedly extrapolating all inactive particles are also minimized by the grouping.

Three integer time variables are assigned to each particle, defining the beginning of its current time step, I_0, its half time step, I_1, and its time increment, I_{dt}. At the beginning of every update, VINE performs a search through the list of all particles to determine the closest future time for which any particle requires a derivative calculation, defined by the condition that

$$I_{\rm next} = \min_i (I_0 + I_{dt}, I_1) \qquad (18)$$

where the minimum is taken over all particles, i, and the resulting value $I_{\rm next}$ defines the time at which the next derivative calculation will occur. Using the value of $I_{\rm next}$, VINE sorts particles onto three lists: particles for which $I_{\rm next}$ matches their end time step time, $I_0 + I_{dt}$, those for which $I_{\rm next}$ matches their half time step time I_1, and those for which neither criterion applies. The three lists correspond to particles requiring 'full' updates to the end of their time step, those requiring 'half' updates at its midpoint, and those which do not require any update at all. Simultaneously, VINE calculates a real valued time step increment, $\Delta\tau$, for use during the integration for each particle. For particles at either their half or full update step, $\Delta\tau$ is identical to the particle's integration step, Δt, otherwise, it defines the difference between the current time and the time of its last update. The values of $\Delta\tau$ are then given to the integrator, all particles are extrapolated to the current time and a derivative calculation is performed.

Two other features of the time step scheme require notice. First, by definition, $I_0 + I_{dt}$ is always greater than I_1 within a single time step. If the value of I_1 is not reset following its half update step, no particle would ever be sorted onto the full update list at all. The problem is compounded by the fact that I_1 cannot be updated until after a full update because of the possibility that the particle move from one time step bin to another. The conflict poses no problem for the leapfrog integrator in practice because no distinction between full and half updates is ever made: the half step update is simply the time at which a new derivative calculation is required. During the extrapolation step, all particles use their value of $\Delta\tau$ directly, while during the update step, active particles simply redefine their step size to twice Δt. On the other hand, the RKF integrator does need to distinguish between particles requiring derivative calculations at their half and full update steps. In order to allow successful searches for particles requiring full updates, VINE replaces the value of I_1 for particles completing their half update steps with the largest allowed integer time, effectively moving it out of the way, so that it can never set the value of $I_{\rm next}$ above. Later, after it completes its full time step, both I_0 and I_1 are updated to appropriate values.

Second, the sequence of updates for different

time step levels proceeds in different order for the leapfrog and RKF integrators. When the RKF integrator is used, particles on the finest time step level become active first at the time of their own half update, then at their full update time. Their full update time corresponds also to the half time step of the next coarser level, so that at that time particles from both levels are active and require derivative calculations. Inactive particles on still higher levels are extrapolated from an earlier time, corresponding to the beginning of their time step, to the current time. As the sequence repeats, particles from as few as one level and as many as all levels may require updates and all fully active particles are updated to the same end time.

Particle updates with the leapfrog integrator occur for only a single time step level at a time, because the time of half time steps on one level do not correspond to the times of half steps on any other level. Both serial and parallel efficiency will be reduced somewhat relative to the RKF because total number of particles updated per cycle is smaller. Also, when particles on coarser levels become active, they advance forward by an increment corresponding to two or more steps on finer levels. This is important because it means that inactive particles on coarse levels must be extrapolated both forward and backwards in time at different points in the sequence as the system is synchronized in preparation for the next derivative calculation. No particle is ever extrapolated more than one half step in either direction however, so the extrapolation remains within the range spanned by the original update.

3. Smoothed Particle Hydrodynamics

The Smoothed Particle Hydrodynamics (SPH) technique for simulating hydrodynamic phenomena was first described by Lucy (1977) and Gingold & Monaghan (1977). Since then, much effort has gone into developing the method (Gingold & Monaghan 1982; Monaghan & Gingold 1983; Monaghan 1985, 1988) and it has become widely used in the study of many astrophysical problems (see Benz 1988, 1990; Monaghan 1992, for reviews).

There are a variety of formulations of the SPH equations, different in various details. In the following, we present the formulation implemented in VINE, and will discuss differences from other formulations and some of the corresponding advantages and disadvantages separately in section 3.3.

SPH solves the hydrodynamical equations in Lagrangian form and can be regarded as an interpolation technique: the positions of the SPH particles combined with an interpolation kernel define the fluid quantities throughout the flow. By default, VINE implements the widely used W4 smoothing kernel defined by Monaghan & Lattanzio (1985) as:

$$W(r_{ij},h_{ij}) = \frac{\sigma}{h_{ij}^{\nu}} \begin{cases} 1-\frac{3}{2}v^2+\frac{3}{4}v^3 & \text{if } 0\le v<1, \\ \frac{1}{4}(2-v)^3 & \text{if } 1\le v<2, \\ 0 & \text{otherwise,} \end{cases} \tag{19}$$

where ν is the number of dimensions and σ is the normalization with values of $2/3$, $10/(7\pi)$ and $1/\pi$ in one, two and three dimensions respectively. Fluid quantities at the position of particle i are then obtained as weighted sums over the properties of all its neighboring particles. For the density, this reads

$$\rho_i \equiv \rho(\mathbf{r}_i) \ = \sum_{j=1}^{N} m_j \ W(r_{ij},h_{ij}) \tag{20}$$

where W is the kernel function defined by equation 19 and m_j are the masses of the, say N, neighboring particles.

The dimensionless separation, $v = r_{ij}/h_{ij}$, between particles i and j, requires the actual magnitude of the separation, $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$, and their characteristic 'smoothing' length scale, h_{ij}, defined as the mean of the smoothing lengths of the two particles:

$$h_{ij} = (h_i + h_j)/2. \tag{21}$$

Thus, with these definitions and the W4 kernel, particles whose separations are $v < 2$ contribute to the summations as 'neighbors' and the influence of particles on each other are symmetric under interchange of indices, an important characteristic required to ensure conservation of momentum and other quantities. All other quantities requiring symmetrization are defined similarly in VINE (see also section 3.3, below).

The choice of kernel used in VINE is made via an external module, so users may substitute an alternate if desired with little or no change to the

rest of the code. The W4 kernel above is second order in the interpolant and has the advantage of being defined on compact support: particles more distant than $v = 2$ do not contribute to the sum. As written (see Vine2, section 3.5), VINE currently defines neighbors as particles with $v < 2$, consistent with the definition appropriate for the W4 kernel. Other choices of kernel might include Gaussian kernels used more commonly earlier in the history of SPH, which are not defined on compact support at all, or other spline kernels, perhaps compactly defined over a different range of separations, v. For these kernels, a new neighbor definition must be made in the code, to define an artificial cutoff or new separation range for the neighbor search. This modification requires a change of only a single line of code, and will therefore be trivial to implement.

3.1. The SPH formulation of the equations of hydrodynamics implemented in VINE

3.1.1. Additions to the Momentum Equation

When gas dynamics are included in a simulation, an additional term must be included to equation 2, which models forces due to pressure gradients and in its ideal form is:

$$\frac{d\mathbf{v}_i}{dt} = -\frac{\nabla p}{\rho} \tag{22}$$

where p and ρ are the pressure and mass density of the fluid, respectively. The SPH formulation used in VINE casts this equation in the form:

$$\frac{d\mathbf{v}_i}{dt} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij} \right) \boldsymbol{\nabla}_i W(r_{ij}, h_{ij}) \,. \tag{23}$$

where $\boldsymbol{\nabla}_i$ means take the gradient with respect to the coordinates of particle i (see e.g. Benz 1990), p is the gas pressure and ρ, the density, is given by equation 20. An additional term, Π_{ij}, appears in equation 23 but has no counterpart in equation 22. Its purpose is described in section 3.1.3 and takes the form of an artificial viscous pressure included to model dissipative effects, without which there is no mechanism to convert kinetic energy into heat due to non-reversible processes such as shocks or viscosity. Alternatively (or in addition), one might include a dissipative term modeled on the viscous terms found in the Navier-Stokes equations directly, however, in its present form VINE does not include such a term.

3.1.2. The energy equation and equation of state

The change in the thermodynamic state of the fluid requires an evolution equation for a state variable corresponding to its internal energy or entropy and Springel & Hernquist (2002) compares the benefits of using one or the other formulation. VINE employs an equation for the specific internal energy of the gas, defined for each particle. In the simplest case of an ideal gas with no external heating or cooling terms, only compressional heating and cooling are important and the equation takes the form:

$$\frac{du_i}{dt} = -\frac{p}{\rho} \boldsymbol{\nabla} \cdot \boldsymbol{v}, \tag{24}$$

where $\boldsymbol{v}$ is the local fluid velocity. Analogously to the momentum equation above, the SPH formulation used in VINE casts this equation with two terms as (Benz 1990):

$$\frac{du_i}{dt} = \frac{p_i}{\rho_i^2} \sum_{j=1}^{N_n} m_j \, \mathbf{v}_{ij} \cdot \boldsymbol{\nabla}_i W(r_{ij}, h_{ij}) + \frac{1}{2} \sum_{j=1}^{N_n} m_j \, \Pi_{ij} \, \mathbf{v}_{ij} \cdot \boldsymbol{\nabla}_i W(r_{ij}, h_{ij}). \tag{25}$$

where the first term corresponds to reversible ('PdV') work. As for the momentum equation, the second term including the viscous pressure Π_{ij} has no counterpart in equation 24. It models the irreversible thermal energy generation produced by the artificial viscosity as it attempts to model shocks or turbulent energy dissipation. A master routine is available for users of VINE to incorporate additional sources or sinks of internal energy via calls to external routines, written to model a specific system. Switches to activate calls to these routines are available to users in a text input file, or can be trivially added to it, which is read in by VINE at run time.

To close the set of equations, an equation of state must also be defined to relate the internal energy, density and pressure to each other. VINE includes options to call several simple equations of state, such as isothermal, isentropic and ideal gases. While the simple choices included will suffice for many problems, we recognize that any set

of equations of state will not be sufficient in general. Therefore, as for energy sources and sinks, a master routine is available for users to incorporate additional equations of state into VINE with minimal difficulty through calls to external routines, activated by switches set in a text input file.

3.1.3. Artificial Viscosity

VINE incorporates an artificial viscous pressure in its momentum and energy equations to model irreversible thermodynamic dissipation from shocks and viscosity. Following standard practice, the viscous pressure includes both a bulk viscosity (the 'α' term) to eliminate subsonic velocity oscillations and a von Neumann-Richtmyer viscosity (the 'β' term) to convert kinetic energy to thermal energy and prevent particle interpenetration in shocks (Monaghan & Gingold 1983). Formally, the expression for Π_{ij} is

$$\Pi_{ij} = \begin{cases} (-\alpha_{ij}\, c_{ij}\, \mu_{ij} + \beta_{ij}\, \mu_{ij}^2)/\rho_{ij} & \mathbf{v}_{ij}\cdot\mathbf{r}_{ij} \leq 0, \\ 0 & \mathbf{v}_{ij}\cdot\mathbf{r}_{ij} > 0, \end{cases} \tag{26}$$

where scalar quantities with both indices i and j are symmetrized values using equation 34 in section 3.3 below. Vector quantities with two indices represent differences, e.g. $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, and μ_{ij} plays the role of the velocity divergence:

$$\mu_{ij} = \frac{h_{ij}\ \mathbf{v}_{ij}\cdot\mathbf{r}_{ij}}{\mathbf{r}_{ij}^2 + \eta^2 h_{ij}^2}\ f_{ij}\,, \tag{27}$$

with $\eta \approx 10^{-1} - 10^{-2}$ to prevent singularities.

In its original form (Lattanzio et al. 1986), f_{ij} is set to unity in equation 27. Balsara (1990, 1995) found that this form gives rise to large entropy generation in pure shear flows, which he suppressed by defining an additional factor f to reduce the contribution selectively in such flow configurations. He defines f_i as

$$f_i = \frac{\left|\left\langle \boldsymbol{\nabla}\cdot\mathbf{v}_i \right\rangle\right|}{\left|\left\langle \boldsymbol{\nabla}\cdot\mathbf{v}_i \right\rangle\right| + \left|\left\langle \boldsymbol{\nabla}\times\mathbf{v}_i \right\rangle\right| + \eta'} \tag{28}$$

with η' preventing divergence once more. VINE allows the user to choose whether or not to allow the factor f to vary or to set it permanently to unity for all particles via a run time option.

The factors α and β in equation 26 are parameters controlling the strength of the artificial viscosity. The best choice for their values depend somewhat on the problem being addressed in a particular simulation (see e.g. Steinmetz & Müller 1993; Lombardi et al. 1999, for some common choices), but values near $\alpha = 1$ and $\beta = 2$ are most commonly used in the literature. VINE allows users to set both values in an input file or, at the user's option, to be set dynamically as conditions warrant at different times or locations in a simulation. The latter setting utilizes a variant of the formulation designed by Morris & Monaghan (1997), where each particle is assigned time dependent viscous parameters, to minimize unwanted viscous dissipation in quiescent flows while retaining good reproduction of the flow properties in shocked regions where it is required. In both standard test problems and in realistic cosmological models of of galaxy clusters, Dolag et al. (2005) found such a time dependent AV formulation to be very useful, yielding much more accurate results compared to the standard formulation with fixed parameters.

Following Morris & Monaghan 1997, each particle is assigned its own value of the viscous coefficient, α_i, which changes in time according to a source and decay equation taking the form:

$$\frac{d\alpha_i}{dt} = -\frac{\alpha_i - \alpha_*}{\tau_i} + S_i. \tag{29}$$

The first term forces the value of α_i to decay asymptotically to a value of α_* on a time scale τ_i, given by

$$\tau_i = \frac{\delta_\alpha h_i}{c_s\sqrt{(\gamma-1)/2\gamma}}, \tag{30}$$

where δ_α is a factor to relate the decay time scale to a more convenient decay length scale describing the distance over which the coefficient decays behind the shock. The form of the decay time is derived by Morris & Monaghan 1997 from the post shock Mach number for strong shocks, combined with the sound speed and smoothing length.

The second term in equation 29 is a source term of the form discussed by Rosswog et al. (2000):

$$S_i = S_0 \max(-f_i(\alpha_{\max} - \alpha_i)\boldsymbol{\nabla}\cdot\mathbf{v}_i, 0) \tag{31}$$

where $\alpha_{\max}$ is a maximum value of the viscous coefficient chosen to ensure that repeated strong compressions or shocks do not increase the viscous parameter to values much larger than those known to yield good results in test problems. The

source function depends on the velocity divergence, so that α_i grows in strong compressions as required, and also includes the Balsara coefficient from equation 28 as suggested by Morris & Monaghan 1997 in order to suppress growth of the source function in shear flows. The scale factor S_0 accounts for equations of state with $\gamma \neq 5/3$:

$$S_0 = [\ln(\frac{5/3+1}{5/3-1})]/[\ln(\frac{\gamma+1}{\gamma-1})], \qquad (32)$$

and is employed to ensure that the peak value of α remains more nearly independent of the equation of state.

As implemented in VINE, each of the parameters α, β, $\alpha_{\max}$, α_* and δ_α may be set at run time by the user with appropriate choice made in a text input file. Both the fixed and time dependent viscosity implementations retain the von-Neumann Richtmyer term for each particle, with the ratio of the two terms β/α also set by the user and fixed to the same value for all particles.

3.2. Variable smoothing lengths

With few exceptions, astrophysical systems exhibit moderate or large density contrasts and ideally one would like to resolve these contrasts as well as possible in simulations. The accuracy of the interpolation of the SPH scheme depends on the number of neighboring particles taken into account in sums like equation 20, which makes high numbers of neighbors (large h values) desirable. On the other hand, large neighbor counts increase the computational effort and decrease the spatial resolution, since only scales larger than h are resolved. These two competing effects lead to a requirement that the number of neighbors should stay roughly constant at a level where correct interpolation is assured without wasting computational resources. Thus the particle smoothing lengths, h, need to vary in space and time, i.e. each particle i gets its own time-dependent smoothing length, $h_i(t)$, which is then integrated according to:

$$\frac{dh_i}{dt} = -\frac{1}{n_d}\frac{h_i}{\rho_i}\frac{d\rho_i}{dt} = \frac{1}{n_d}\, h_i\, \boldsymbol{\nabla}\cdot\mathbf{v}_i\ , \qquad (33)$$

where n_d is the number of dimensions and the continuity equation has been used to replace the time derivative of the density (Benz 1990). If the integration of equation 33 leads to a neighbor count outside a given range, an exponential correction term pushes $|\dot{h_i}|$ to greater values, such that more/less neighbors are found on the next step if too few/many were found on the current step. VINE attempts to keep the neighbor count within the range [30, 70] in 3D and [10, 30] in 2D. These ranges depend somewhat on the total number of particles (see Lombardi et al. 1999, and references therein) and on the physics itself (Attwood et al. 2007), and so are defined as parameters that can be changed when the code is compiled.

The error introduced into the interpolation scheme by allowing for varying smoothing lengths is second order in h (Monaghan 1985; Hernquist & Katz 1989). Since SPH with constant smoothing lengths is also only accurate to second order in h with the kernel in equation 19 (see e.g. Benz 1990), the addition of variable smoothing lengths imposes no additional restriction. However, variable smoothing lengths formally introduce an additional term in the interpolated forms of the gradients of functions, as these forms involve gradients of the kernel. These additional terms can be neglected in many simulations (see Evrard 1988), but doing so may cause spurious entropy generation in some cases Nelson & Papaloizou 1994; Springel & Hernquist 2002; Price & Monaghan 2004. In its present form, VINE neglects the additional terms, but modifications to the code to implement them are planned for the future.

3.3. Symmetrization of quantities in SPH

The description of the SPH equations so far has only dealt with the specific form implemented in VINE. Other formulations exist (e.g. Gingold & Monaghan 1982; Evrard 1988; Hernquist & Katz 1989), but a thorough review of even the most common is beyond the scope of this paper. We suggest Thacker et al. (2000) for a more detailed discussion of the alternatives, and limit ourselves here to an outline of a few of the major points regarding the differences and why we use the scheme implemented in VINE.

Most differences between the SPH implementations come from the way force symmetry between pairs of particles is handled. For momentum conservation, symmetry between interchange of indices in the pairwise force calculation between

two particles is required. As noted above, VINE uses the arithmetic mean of quantities defined for each particle to produce the symmetry:

$$q_{ij} = \frac{q_i + q_j}{2}, \tag{34}$$

where q is some quantity intrinsic to a particle such as smoothing length or sound speed. Specifically for the calculation of the kernel and its gradient, VINE uses a symmetrized smoothing length, h_{ij}. Alternatively, some implementations derive force symmetry from a symmetrized kernel. Rather than defining $W(r_{ij}, h_{ij})$, these implementations define $\tilde{W}_{ij}$ as

$$\tilde{W}_{ij} = \frac{1}{2}W(r_{ij}, h_i) + \frac{1}{2}W(r_{ij}, h_j), \tag{35}$$

as originally suggested by Hernquist & Katz (1989) and used e.g. by Dave et al. (1997); Carraro et al. (1998); Lia & Carraro (2000); Springel et al. (2001) and Wadsley et al. (2004). Other symmetry differences originate in the form of the momentum equation and energy equations where, rather than calculate a sum of pressure forces from each particle in a pair, a geometric mean is taken.

The choice of symmetrization has direct implications for the neighbor search as well. To assure force symmetry, neighboring particles i and j must be determined to be mutual neighbors. Some codes use range search techniques (see e.g. Steinmetz 1996; Springel et al. 2001, note that only the serial version of Gadget-1 uses this technique). In this case, all particles j with $|r_i - r_j| \leq \eta\, h_i$ are used as neighbors of i, with $\eta > 1$ and typically $\eta \approx 1.3$. This method does not guarantee that both particles i and j will find each other as neighbors. Using $h_{ij} = (h_i + h_j)/2$ for the neighbor search in each individual comparison of two particles assures by construction that the mutual neighbor property is satisfied for all pairs i, j. The higher computational cost (if any) required to complete searches is more than offset by the fact that symmetrizing the kernels through equation 35 leads to evaluating the kernel function (or its gradient) twice, while VINE only needs to evaluate them once.

3.4. Additional time step criteria for SPH simulations

Two additional time step criteria are required in SPH simulations. First the Courant-Friedrichs-Lewy condition must be satisfied:

$$\Delta t_{\rm CFL}^{n+1} = \tau_{\rm CFL} \frac{h_i}{c_i + 1.2\,(\alpha_i\, c_i + \beta_i\, h_i \max_j \mu_{ij})} \tag{36}$$

where α_i and β_i are the artificial viscosity parameters and c_i is the sound speed for particle i, and μ_{ij} is defined by equation 27 with the maximum taken over all neighboring particles j of particle i. The form of equation 36 is that suggested by Monaghan (1989), who recommend values of $\tau_{\rm CFL} \approx 0.3$ for good results.

Secondly, VINE requires that the smoothing length of each particle must not change too much in one timestep:

$$\Delta t_h^{n+1} = \tau_h \frac{h_i}{\dot{h}_i}\,, \tag{37}$$

where τ_h is a tuning parameter. We set $\tau_h \approx 0.1 - 0.15$ in order to ensure that particles encountering strong shocks require several timesteps to pass entirely through the interface. For a gas with a ratio of specific heats, $\gamma = 5/3$ for example, the density enhancement across a strong shock will be a factor of four, corresponding to a smoothing length change of $4^{1/3} \approx 1.6$ in 3D. With a density jump of the severity and suddenness experienced in a shock, the timestep restriction of equation 37 will become important and the particle's timestep will decrease, allowing better resolution of the physical conditions of the shock interface. With $\tau_h = 0.15$ for example, a particle will require at least $\sim 4 - 5$ timesteps to contract fully to the post-shock condition.

The criterion in equation 37 is also beneficial in situations where the spatial gradient of neighbors is large, as it may be for particles on the surface of an object. Small changes in the particle's smoothing length can then lead to very large changes in its neighbor count and ultimately to large oscillations between far too many and far too few neighbors, from one timestep to the next. A timestep restriction dependent on the smoothing length variation restricts such particles to correspondingly smaller timesteps, so that such oscillations do not develop.

Preventing or damping such oscillations is important both because of the comparatively high cost of computing the evolution of such boundary particles, but also because they can lead to oscillations in the physical quantities such as pressure

forces, enhancing oscillations further as particles experience large intermittent kicks. Although certainly more fundamental in a physical sense, in a numerical sense quantities such as density are actually derived from numerically relevant smoothing length of a particle. Therefore, rather than limiting the change in derived quantities such as density, VINE limits the change to the particle's smoothing length, as a more direct throttle on unphysical behavior.

4. Gravity

The most costly calculation in nearly any particle simulation including gravity is the computation of the gravitational forces of the particles on each other. This is due to its long range nature, which couples all particles in the system to each other. Modeling many astrophysical systems requires forces accurate only to $\sim 0.1 - 1\%$, due to the fact that other errors already present in the various approximations made in the model are of similar or greater magnitude. For such systems, an approximate solution will be not only acceptable, but much to be desired if it can be obtained much more quickly than an exact solution. For other models, e.g. star clusters, highly accurate forces must be determined for accurate evolution in spite of their cost; an approximate solution will be useless. VINE implements both approximate and exact methods for calculating mutual gravitational forces of particles on each other. In this section, we provide an overview of our implementation of these alternatives. We refer the reader to Vine2 for a lower level description of the detailed methods. Users can choose from among the available options at run time an appropriate choice in a text input file.

4.1. Exact gravitational forces obtained from direct summation

The most naive approach to calculating the magnitudes of interactions of particles on each other is simply to calculate directly a sum of the terms due to every particle on every other particle:

$$\boldsymbol{a}_{\rm i} = -\sum_{j}^{N_p} \frac{G m_j}{|\boldsymbol{r}_i - \boldsymbol{r}_j|^2} \hat{\mathbf{r}}_{ij} \qquad (38)$$

where $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ are the positions of particles i and j, $\hat{\mathbf{r}}_{ij}$ is the unit vector connecting them and m_j is the mass of the j'th particle. This calculation is extremely expensive, requiring $\mathcal{O}(N_p^2)$ operations for every time step. VINE includes two alternatives for computing the sum in equation 38 for every particle. The sum can either be computed directly by the processor, or it can be computed instead by special purpose hardware, so called 'GRAPE' coprocessors, described in section 4.3 below.

4.2. Approximate gravitational forces obtained from tree based sorting

When physical models do not require or do not allow for exact inter-particle gravitational force calculations, as will be the case for so called 'collisionless' systems, approximate forces may become a desirable alternative if they speed up the required computations while still retaining sufficient accuracy. For gravitational force calculations, approximate forces can be obtained by aggregating the contributions of distant particles into a single interaction, for which all of their individual contributions together can be approximated as being due to a sum of multipole moments, perhaps truncated to some low order.

The difficulties in this approach lie in establishing how big each aggregate can be before errors in the force become overwhelming, and in sorting through all of the possible nodes, making certain that every particle is included in the force calculation for every other particle exactly once, either as an individual or as part of an aggregate. The most common and most general purpose method for obtaining such approximate forces is to organize the particles into a tree data structure, and then use tree nodes as proxies for groups of particles. By examining successive nodes in the tree, all of the particles contained in that node can be either qualified as interactors, or the node can be opened and its children examined for acceptability instead. Fully traversing the tree produces a list of nodes determined to be acceptable for interaction with a given particle, and a list of atoms (single particles), for which an exact computation is required. Using a tree to determine a list of neighbors or acceptable nodes reduces the overall computational effort to $\mathcal{O}(N_p \log N_p)$.

The challenge in making the use of a tree efficient are first to choose an efficient method to traverse the tree, and second, to choose an efficient

method to decide which tree nodes are acceptable as is, and which must be separated into their constituent parts to be examined in more detail. We will describe both the traversal strategy and the node acceptability criteria used in VINE in detail in Vine2. For purposes here, it will be sufficient to describe qualitatively the criteria used to determine node acceptability.

In order to calculate an accurate gravitational force due to some node which defines a particle distribution, the error it contributes to the total force on a particle must be small. Mathematically, for a multipole expansion to converge at all, this condition translates to the physical statement that the particle on which the force acts must be remote from the mass distribution defining the node (see e.g. Jackson 1975, ch. 4). In addition, if the multipole expansion is truncated rather than continuing to infinite order, as will be desirable in an approximate calculation in order to save time, errors corresponding to the neglected contributions from the higher order terms must also be small.

VINE implements three runtime selectable options for determining the acceptability of a node to be used in the gravitational force calculation, each based on a different implementation of the convergence radius of a multipole expansion and of limits on the errors due to series truncation. Following common practice, we call each of these options 'Multipole Acceptance Criteria', or MACs. Each MAC includes a user settable parameter 'θ' by which the accuracy in different problems may be tuned to the most suitable value for that problem, but the interpretation given to θ is specific to each MAC.

The first and simplest is the 'geometric' MAC:

$$r_{ij}^2 > \left(\frac{h_j}{\theta} + h_i\right)^2 = R_{crit}^2, \qquad (39)$$

where h_i and h_j are the physical sizes of the particle and node, respectively and r_{ij}^2 is the square of their separation. The accuracy parameter θ takes a value between zero and one, and parameterizes the minimum acceptable distance at which a node may be used in the gravity calculation. Alternatively, by switching the positions of θ and R_{crit} and setting $h_i = 0$ it may be interpreted instead as the tangent of the angle subtended by the node on the 'sky' as seen by the particle. We incorporate the size of the particle, h_i, into the MAC in order to ensure that the condition is satisfied for all locations inside it, and to allow a generalization of the MAC to be used for groups of particles taken together (see Vine2 for additional details). Errors due to truncation of the multipole expansion are implicitly assumed to be small, because each higher order term is diminished by an additional power of the separation, r, appearing in the denominator of each multipole term.

Second, we implement the absolute error criterion of Salmon & Warren (1994) (the 'SW' MAC), who derive explicit limits on the errors due to series truncation. A node, j, is acceptable for use in the gravity calculation if

$$r_{ij}^2 > \left(h_i + \frac{h_j}{2} + \sqrt{\frac{h_j^2}{4} + \sqrt{\frac{3\mathrm{Tr}\,\boldsymbol{Q_j}}{\theta}}}\right)^2 \qquad (40)$$

where θ is a value defining the maximum absolute error in the acceleration that a single node may contribute to the sum and $\boldsymbol{Q_j}$ is the quadrupole moment tensor for node j. When the quadrupole moment is zero, this criterion reduces exactly to the geometric MAC above with its θ defined to be unity, defining a simple separation criterion. We note here also that the original formulation of Salmon & Warren (1994) includes the possibility of including the size of the particle, h_i, directly in the term under the square root. Instead, we choose the form in equation 40 in order to make possible a single calculation of the MAC definition for each node prior to any tree traversals.

Finally, we implement the MAC discussed in Springel et al. (2001), which we refer to as the 'Gadget' MAC, and which uses an approximation for the truncation error of the multipole expansion at hexadecapole order to define an error criterion. A form using an octupole order error formulation can be derived, but is computationally costly for modern computers because it contains a square root, and is therefore not used. For the relative error in the acceleration contribution of the node, compared to the total acceleration at the last time step their criterion is:

$$r_{ij}^6 > \frac{M_j h_j^4}{\theta |\boldsymbol{a}_{\mathrm{old}}|} \qquad (41)$$

where the gravitational constant implicitly present in the formula is set to $G = 1$, $\boldsymbol{a}_{\mathrm{old}}$ is the value

of the acceleration for a particle at the last time it was calculated and θ is a dimensionless maximum relative error in the acceleration to be allowed to any acceptable node. Because it requires a previous value of the acceleration, equation 41 cannot be used for the first calculation. Instead, we use the geometric MAC of equation 39 with its accuracy parameter set to $\theta = 0.5$. Also, equation 41 may not ensure that particle and node do not overlap in space, violating the separation condition required for convergence of the multipole summation. Therefore, when the Gadget MAC is selected, we also require that equation 39, with its parameter set to $\theta = 1$, is satisfied.

4.2.1. The acceleration calculation

To compute the gravitational acceleration due to each entry on the list of acceptable nodes on some particle, i, users of VINE may choose one of two options. First, they may choose to sum the contributions from acceptable nodes and atoms using a multipole summation truncated at quadrupole order and computed on their general purpose 'host' processor or, alternatively, they may instead choose to compute the summations to monopole order using a GRAPE coprocessor, if one is available.

For the former, VINE sums the multipole series for each acceptable node using the unreduced quadrupole moment formulation described in Benz et al. (1990). Mathematically, the acceleration on particle i due to node j is:

$$\begin{aligned} \boldsymbol{a}_{\rm i} &= M_j f(r)\boldsymbol{r} + \frac{f'(r)}{r}\boldsymbol{Q}_j \cdot \boldsymbol{r} \\ &+\frac{1}{2}\left[\frac{f''(r)}{r^2}\boldsymbol{r}\cdot\boldsymbol{Q}_j\cdot\boldsymbol{r}\right. \\ &\left. + \frac{f'(r)}{r}\left(\mathrm{Tr}\,\boldsymbol{Q}_j - \frac{\boldsymbol{r}\cdot\boldsymbol{Q}_j\cdot\boldsymbol{r}}{r^2}\right)\right]\boldsymbol{r} \end{aligned}$$

where the function $f(r) = Gr^{-3}$, $\boldsymbol{r}= \boldsymbol{r}_j - \boldsymbol{r}_i$ and $\boldsymbol{Q}_j$ is the quadrupole moment tensor for node j. No specific gravitational softening (see section 4.4) is required in equation 42 because the criteria used to determine the list of atoms ensures that no nodes will require softening. Atoms are handled independently and contribute only a monopole term, softened according to the options discussed below. The acceleration due to all atoms and nodes together are summed to obtain the gravitational acceleration acting on a particle, i.e. the right hand side of equation 2.

4.3. GRAPE hardware

As noted above, VINE includes options to calculate gravitational forces using GRAPE coprocessors if they are available, either in approximation or exactly. 'GRAPE' hardware (for **G**r**A**vity **P**ipelin**E**) to accelerate N-body calculations has been developed by several collaborations in Japan (Sugimoto et al. 1990; Fukushige et al. 1991; Ito et al. 1991; Okumura et al. 1993; Makino et al. 1997; Kawai et al. 2000; Makino et al. 2003; Fukushige et al. 2005, see also Makino & Taiji 1998 for a review), and has been used both by them and by many others throughout the astronomical community (see, e.g., Naab & Burkert 2003; Athanassoula 2003; Naab et al. 2006a; Merritt & Szell 2006; Portegies Zwart et al. 2006; Berczik et al. 2006). This hardware is attractive to users because it is a system of specialized computer chips designed to perform a summation of $1/r^2$ force calculations very quickly using a highly parallelized pipeline architecture. These chips are combined onto a processor board which also hosts all other other necessary functional units, including memory, I/O controller, etc, and which is then attached to a host computer, usually a desktop workstation. which performs the rest of the calculations required in a simulation.

Two types of GRAPE boards exist, differentiated by even or odd version numbers. GRAPE boards with odd numbers have a less accurate numeric format, so that the relative error of the total force on a particle is of the order of 2% for GRAPE-3 (Okumura et al. 1993) and 0.3% for GRAPE-5 (Kawai et al. 2000). For collisionless systems, this imposes no problem for the time evolution, as the errors are uncorrelated (Makino et al. 1990). GRAPE systems with even version numbers, such as GRAPE-2 (Ito et al. 1991), GRAPE-4 (Makino et al. 1997) and the most recent, GRAPE-6 (Makino et al. 2003) have higher accuracy in their internal numeric formats and were designed for simulating of collisional systems like globular clusters.

VINE supports GRAPE-3, GRAPE-5 and GRAPE-6 boards, the latter in both a full and reduced size 'MicroGRAPE' form, also known as

'GRAPE-6A' (Fukushige et al. 2005). The code can use GRAPE boards for direct summation of forces as well as a component in an approximate tree based gravity calculation. We describe the details of this approach in Vine2 after the tree itself has been described in detail.

4.4. Softening the forces and potential

Simulations of physical systems depend on a choice to assume that the system is either 'collisional' or 'collisionless', corresponding to the statement that the evolution of the system as a whole is or is not strongly affected by the outcomes of interactions between individual particles ('collisions'). In hydrodynamic systems for example, collisions between individual atoms or molecules matter only in the aggregate, as a pressure. In N-body systems of galaxies for example, where individual particles may in fact be stars rather than atoms, a similar statement can be made if the two body relaxation time is long compared to the expected simulation time or the lifetime of the system.

Even though the underlying physical systems may be collisionless in the sense that no particular interaction affects the result, simulations of them may not be because the number of particles used in the simulation is typically many orders of magnitude smaller than the actual number of bodies in the real system. Particles in these simulations are actually meant to represent an aggregate of some very large number of physical particles. Actual collisions between them, as individuals, are therefore unphysical. In order to recover the collisionlessness of the physical system, forces between particles must be 'softened' in some manner or, in other words, the particles must be provided with an internal density distribution consistent with their interpretations as aggregates of many particles rather than as a distinct entity.

In practice, softening is achieved by modifying the gravitational potential on small scales to avoid the pure $1/r$ form and the associated numerically infinite forces at small separations. The softened form actually defines a mass density distribution for each particle, so that they include an assumption of some spatial extent, rather than that they are point like objects. Using softening, the artificial close encounters between N-body particles are suppressed and the particles evolve under the action of the global gravitational potential created by all particles.

4.4.1. Forms of gravitational softening in VINE

A long standing debate pervades the astrophysical literature concerning how gravitational softening should be implemented in particle simulations (see e.g. Merritt 1996; Bate & Burkert 1997; Romeo 1997; Athanassoula et al. 2000; Dehnen 2001; Nelson 2006, and references therein for detailed discussions). No form of softening so far proposed exists which is free of defects and some care must be taken to ensure simulation results do not depend on the implementation of softening. In practice, two alternatives are common; first to use 'Plummer softening' and second to use a spline based kernel, perhaps also with a space and time dependent length scale. VINE implements both options, via a user selectable switch.

Plummer softening defines the density function of each particle to be a Plummer sphere, so that the force on particle i due to particle j, at a distance $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$, becomes

$$\mathbf{F}_i = -\frac{Gm_i m_j}{r^2 + \epsilon^2}\hat{\mathbf{r}}_{ij}, \tag{42}$$

where $\hat{\mathbf{r}}_{ij}$ is the unit vector connecting particles i and j. The potential is defined similarly as

$$\Phi = -\frac{Gm_j}{(r_{ij}^2 + \epsilon^2)^{1/2}}, \tag{43}$$

where ϵ is a softening length scale. Plummer softening was originally motivated by Aarseth (1963), who used it in simulations of clusters of galaxies. Its most compelling advantage is that it is simple to implement and computationally inexpensive. Its major drawback is that it never converges to the exact Newtonian potential at any separation.

Spline based softening defines the density function of a single particle using a predefined kernel that extends over some finite region in a manner not unlike that used to derive hydrodynamic quantities in SPH. Although in principle any kernel could be used, VINE implements the kernel defined in equation 19, used for the SPH interpolations. This kernel has the advantages that it has compact support, i.e. for $r > 2h$ it takes the exact Newtonian form, that for hydrodynamic simulations with smoothing and softening set equal,

pressure and gravitational forces are nearly equal and opposite on small scales (Nelson 2006), and that for a given number of particles, errors in the force calculation are smaller than that of many other possible kernels (Dehnen 2001).

To specify the force on particle i due to particle j, we apply Gauss's law and integrate over the kernel density distribution to obtain the fraction of the source particle's (j) mass enclosed by a sphere with radius equal to the separation between them:

$$\begin{aligned} \hat{m}(r_{ij}) &= 4\pi \int_0^{r_{ij}} \rho(v) v^2 dv \\ &= 4\pi \int_0^{r_{ij}} W(v, h_{ij}) v^2 dv \,, \end{aligned} \quad (44)$$

where the density, ρ, is replaced by the softening kernel W in the second equation, and v is as defined in section 3. In 2D, a similar form for $\hat{m}_j$ can be defined but leads to a finite, *non*-zero force at zero separation (Nelson 2006). VINE instead modifies the form of the kernel in 2D to that proposed by Nelson 2006, to avoid the inconsistency. Given the modified definition of the source particle's mass, the force and potential are defined by the equations for Newtonian gravity:

$$\mathbf{F}_i = -\frac{G m_i \hat{m}_j}{r_{ij}^2} \hat{\mathbf{r}}_{ij} \quad (45)$$

and

$$\Phi = -\frac{G \hat{m}_j}{r_{ij}}. \quad (46)$$

Note that as two particles come closer (without decreasing their softening lengths), the force decreases to zero because the mass enclosed decreases to zero, and that the force is antisymmetric with respect to interchange of i and j. At great distances, $\hat{m}_j$ becomes m_j and the exact Newtonian form is recovered.

Three softening variants are available in VINE, via options specified at runtime. First, Plummer softening may be selected with a fixed softening length ϵ for all particles. When the GRAPE option (see section 4.1) is selected for gravitational force calculation, this is the only option available due to hardware constraints. Second, either 'fixed' or 'variable' kernel softening may be selected, with the latter option affecting only the treatment of SPH particles. If the fixed option is selected, each SPH particle i is softened using a single softening length $h_i = \epsilon$, specified at run time for all particles. After the gravity calculations are completed, smoothing lengths are reset to their locally defined, spatially and temporally variable values so that later hydrodynamic calculations may proceed. If the variable option is selected, the individual (and time varying) smoothing lengths of each particle are used as individual softening lengths. In both the fixed and variable kernel softening options, all N-body particles use the kernel and their predefined (fixed) softening lengths h_i. This alternative allows several species of N-body particles to be included, each with their own (possibly different) softening length.

Depending on the softening selection, either terms from equation 42 or equation 45 (in the form of accelerations) are summed over all particles derived from the tree traversal to obtain the gravitational acceleration acting on a particle, to be in the right hand side of equation 2.

4.4.2. Spline softening and the connection with SPH

Calculation of gravitational forces between neighboring SPH particles requires the identification of equation 19 as a density distribution, a different and stronger assumption that its use as an interpolation kernel in non-self gravitating SPH. Two important consequences arise from the identification. First, although there is no requirement that the hydrodynamical smoothing length and the gravitational softening lengths are equal, large force imbalances may develop if they are not, due to the different assumptions about the mass distribution within a single particle. The imbalances require careful consideration because they are consequences of the numerical assumptions, not of the physical systems, but can catastrophically change the outcome of a simulation. We refer the reader to Bate & Burkert (1997) and Nelson (2006) for more detailed discussions, but note the main conclusion of both is that more favorable outcomes are obtained when gravitational softening and hydrodynamic smoothing lengths are set to be equal.

Secondly, equation 44 will introduce fluctuations into the potential energy of the system (and of individual particles) when it is used with variable smoothing lengths because the change in mass

distribution within a particle is not accounted for in the potential when its smoothing length changes. In most cases, the variation will be small because it comes only from SPH neighbor particles, rather than from the entire mass distribution. However, in some cases it may be an important local or global effect, such as in uniformly collapsing or expanding media, or in cases when a small subregion collapses by many orders of magnitude. This violation of energy conservation due to smoothing length variations can be estimated as (Benz 1990):

$$dH = -G \sum_{i<j} m_i m_j \frac{\partial \Phi}{\partial h_{ij}} dh_{ij}. \qquad (47)$$

In principle this formula could be used to correct the potential energy of SPH particles (but not their accelerations), but we have not included it in VINE because it requires an additional search for neighbor particles, with its associated computational cost. We also note that a conservative form of softening has recently been developed by Price & Monaghan (2007), but this form has yet to be implemented in VINE.

5. A third particle species: 'star' particles

Astrophysical systems frequently include multiple physical components with substantially different characteristics, embedded within and interacting with each other. In this context, a common characteristic in astrophysical systems is the interaction between a comparatively low density gaseous medium and one or more stars. Such systems include, for example, a single star surrounded by a gaseous circumstellar disk (Nelson et al. 1998, 2000), or cloud of gas in which one or many stars form during the evolution (Bonnell et al. 2006). For modeling simplicity, each component may require separate treatment in order to capture essential physical phenomena peculiar to only one without distorting the evolution of another, and include explicit coupling terms to model their interactions.

In addition to the capability to evolve systems simulated in the purely N-body and SPH particle frameworks already described, VINE also includes the capability to model a third species of particles, similar to N-body particles, but with additional characteristics not present in the other types. This species is designed to be used in models requiring 'stars' or other similar point masses in astrophysical systems. VINE evolves star particles using the same integration method used for all other particles (section 2), and with the same time step criteria, though different coefficients may be assigned. They interact gravitationally with all other particles and with each other according to a softened Newtonian force law, as described in section 4.4. Because they may be many orders of magnitude more massive than the other types of particles, they are not included in the approximate, tree based force calculation, but instead take advantage of the fact that there will typically be a very small number of such particles to obtain forces via direct summation.

The principle difference between VINE's implementation of star particles and the simple N-body particles already discussed is their ability to 'accrete' particles of the other two types, should their trajectories bring them into close proximity. If activated by a switch set by the user at run time, VINE makes a check to determine whether any particles have moved to within one smoothing length, h_*, of each star after each timestep. When individual time steps are used, both the accretor and accreted particles must be at the end of their time step, in order to maintain simplicity in the code and the integration scheme. When a particle is found, it is removed from the simulation and its mass and momentum are added to the star. In order to conserve the center of mass of the system, the star particle is artificially moved to the common center of mass of the pair. Similarly, VINE calculates the angular momentum of the star and fluid particle around their common center of mass, then adds it to the star as an internal 'spin', in order for any later accounting of the system's total angular momentum to be conserved.

6. Cosmological Expansion

For simulations of large scale structure formation in the universe as well as for galaxy formation simulations in a cosmological context, the underlying cosmological model has to be taken into account. The expansion of the universe on these scales plays an important role and thus the equations of motion have to incorporate corresponding terms. It is common practice to perform such

simulations in comoving coordinates. Some authors have chosen to use yet different coordinates for the integration, which were especially adapted for the context of cosmological simulations, e.g. Springel et al. (2001). For VINE, we wanted to minimize of modifications necessary to take cosmological expansion into account in order to keep the code as modular as possible. So we did not follow the above approach and use comoving coordinates for the integration. The modifications are then limited to changing the equations of motion. The necessary terms are incorporated only into the leapfrog integration scheme (see section 2.2). VINE is not designed to use the Runge-Kutta-Fehlberg integrator described in section 2.1 for cosmological simulations. This, however, is not a severe limitation as practically all other cosmological simulation codes make use of the leapfrog scheme because it only requires one force calculation per time step.

We adopted an implementation where the only modification is in the equation for the velocity update in the leapfrog scheme, see equation 7, following Efstathiou et al. (1985) and Bode & Ostriker (2003). For completeness, we write again the full set of equations for the leapfrog integrator with cosmological expansion:

$$\mathbf{x}_{n+1/2} = \mathbf{x}_n + \frac{1}{2}\mathbf{v}_n \Delta t_n \tag{48}$$

$$\mathbf{v}_{n+1} = \frac{1 - H_{n+1/2}\Delta t_n}{1 + H_{n+1/2}\Delta t_n}\,\mathbf{v}_n + \frac{a_{n+1/2}^{-3}}{1 + H_{n+1/2}\Delta t_n}\,\mathbf{g}_{n+1/2}\Delta t_n \tag{49}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_{n+1/2} + \frac{1}{2}\mathbf{v}_{n+1}\Delta t_n \tag{50}$$

where a is the scale factor, $H = \dot{a}/a$ the Hubble parameter and $\mathbf{g}$ is the gravitational acceleration. Indices n, $n+1/2$ and $n+1$ indicate quantities at time t_n, $t_{n+1/2}$ and t_{n+1}, respectively, and $t_{n+1} = t_n + \Delta t_n$.

The quantities $a(t)$ and $H(t)$ need to be computed for every time step according to the cosmological model used. Their calculation is inexpensive and does not significantly increase the total time required for the integration scheme.

7. Boundary Conditions

In many contexts, simulations of entire physical systems may be possible for many astrophysical systems of interest. Other systems may be studied only as some small subset of a larger whole, e.g., a cosmological simulation which naturally models only a small portion of the universe. In either case, a complete model requires that conditions at the boundaries of the computational domain be specified in order to model the influence of matter from outside it.

For particle based simulations of entire systems, 'free' boundary conditions require no special treatments because the fluxes of material, momentum and energy are carried by the particles themselves. Particles feel perturbations only from other particles already present in the simulation, not from any specific boundary, and move into adjacent to empty regions as conditions warrant. Otherwise, such regions require no computational effort. More complicated systems require active management of particles near boundaries, with specific treatments tailored to specific problems.

For the example cosmological simulations above, the relevant treatment will be of a volume of space surrounded by an infinite series of other, identical volumes, replicated in succession at greater and greater distances in each direction from the original. VINE includes a module to implement periodic boundary conditions in this context and we describe it below. Being much rarer in astrophysical contexts, we have chosen not to implement reflecting boundaries, for which material approaching some surface is repelled with identical velocity but opposite sign, although a template routine has been included for this purpose.

7.1. Periodic Boundaries

When periodic boundary conditions are employed, all matter resides inside a simulation box. The boundary conditions then imply that calculations of physical quantities, such as gravitational forces, gas densities, etc. are carried out as if the box were surrounded in every direction by a series of identical replicas, extending to infinite distance in all directions. Particles whose trajectories pass through one of the box's boundaries, entering one of the replications, are artificially restored to the original box on the opposite boundary. There-

fore, after every position update, VINE performs a check for particles which have moved outside of the box. If any such particles are detected, VINE adds or subtracts the length of the box to the appropriate position component of those particles, effectively reinserting them on the opposite side of the volume from the one from which they exited. Velocities are unchanged.

7.1.1. Gravitational Periodic Boundaries

Various treatments of gravitational periodic boundary conditions in particle evolution codes have been discussed in the literature. Most are based on the Ewald method (Ewald 1921) and we refer readers to Hernquist et al. (1991) for the application to astrophysical particle simulations (note however that their eq. 2.14b is missing a factor $|\mathbf{r} - \mathbf{n}L|$, see Klessen 1997). The Newtonian potential of the particles is replaced by the potential of an infinite, periodic replication of the particles. Using Ewald's method, the resulting slowly converging sums for the potential and force are replaced by two pairs of partial sums. One contains the short range terms, and the other the long range terms. Since the short range summations are carried out in real space and the long rage sums in Fourier space, both converge quickly and can thus be cut off after a small number of terms.

Most practical implementations use a modified force law for computing the forces from any node or particle on the interaction list obtained from traversing a tree (see e.g. Dave et al. 1997). In conflict with such implementations, GRAPE hardware can only compute Plummer's force law without any modifications, making periodic boundary problems impossible if users wish to use GRAPE hardware. In order to preserve the possibility of the use of GRAPE hardware in VINE in all cases, we have therefore chosen an alternate implementation of Ewald's method (Klessen 1997). Instead of modifying the gravitational force law for all interactions inside the boundary box, the forces from matter inside the simulation box is computed as if no boundary was present, then all particles get a correction force to account for the contributions from neighboring boxes.

To determine the correction, VINE maps the particles onto a grid using the CIC (Cloud-In-Cell) scheme (see e.g. Hockney & Eastwood 1981), creating a density distribution on the grid, which is then Fourier transformed. The result is convolved with a Green's function describing the correction and then transformed back into real space. Finally, the forces and potential are mapped from the grid back onto the particles, giving every particle the desired correction of the force and potential at the particle's location. Since the forces from particles inside the box are still computed using the tree or GRAPE-tree combination as usual, it is necessary to use a modified Green's function on the grid. This function is the usual Green's function for Ewald's method minus the contribution of matter inside the simulation box. For a detailed description of the method, we refer the reader to Klessen (1997).

The computational cost of the calculation of the correction terms is governed first by the grid resolution and second by the speed of the FFT algorithm used. Experiments show that sufficently accurate forces can be obtained if the linear density of grid cells is at least twice that of the particles. For example, if e.g. 64^3 particles are used, the grid should at least have a resolution of $> 128^3$. Most publically available and vendor supplied numerical libraries include some variant of fast, parallel FFT algorithms. Rather than writing multiple variants of our Ewald code for each of these libraries, we link to the FFTW library (Frigo & Johnson 2005), commonly available on most computing platforms, in order to retain both maximum portability for VINE and to realize high performance on all platforms.

7.1.2. SPH Periodic Boundaries

Calculations of hydrodynamic quantities are always local in the sense that only neighboring particles contribute to any given particle's hydrodynamic quantities. No complex algorithms, such as Ewald's method for gravity, are required to determine forces due to particles in neighboring replications of the simulation box. The only change that must still be made is to modify the definition of the distance between pairs of particles, so that particles located near a boundary see neighboring particles from both sides of that boundary.

The required modification utilizes the fact that no particle can be more distant from another than one half of the box size in any direction. If it were, its ghost particle in a neighboring box would in-

stead be chosen as neighbor because its separation in that direction was smaller. We can therefore define the separation in each direction as

$$\delta x_{ij} = x_i - x_j - L \ \mathrm{nint}\left(\frac{x_i - x_j}{L}\right) \qquad (51)$$

where L is the box length and '`nint`' stands for the 'nearest integer' function and takes the value of zero or ± 1, depending on the value of its argument. With this definition, and accounting for the periodicity of the box, the separation of a pair of particles can be calculated simply as the difference between their natural coordinates, plus an extra term which reduces to zero when the natural separation is small. If the magnitude of their natural separation is larger than $L/2$, the box length will be added or subtracted as appropriate. Combining the separations in each of the coordinate directions yields a distance. The computational cost of the extra operations are quite small, and no additional code infrastructure is required to handle the existence and storage of actual ghost particles. No additional infrastructure is required for neighbor searches either, since equation 51 is trivially generalizable to calculations involving tree nodes as well. After neighbor identification and distance calculations are complete, computations of all hydrodynamic quantities, whether involving the distance metric directly or indirectly, through the SPH kernel (e.g. densities or velocity and pressure gradients), proceed as in the case of free boundaries.

The definition of separation in equation 51 has one minor side effect, that any particle so large that it is a neighbor of both some other particle contained in the simulation volume *and* one or more of that particle's duplicates in neighboring volumes will find only one instance of that particle. Its duplicates will not be found. We consider this situation to be extremely unlikely for any simulation containing more than a few tens of particles, and so neglect the possibility entirely.

8. Test Simulations

In this section we present various tests of the code, demonstrating the capabilities of VINE for SPH simulations as well as for N-body simulations.

8.1. Adiabatic collapse of a cold gas sphere

Since Evrard (1988) the adiabatic collapse of a cold, initially isothermal gas sphere under its own gravity has been a widely used test case for SPH codes, see e.g. Hernquist & Katz (1989); Steinmetz & Müller (1993); Hultman & Källander (1997); Carraro et al. (1998); Springel et al. (2001); Thacker et al. (2000). Adopting a similar setup as these authors, we simulate the collapse of a spherically symmetric gas cloud with density profile

$$\rho(r) = \frac{M}{2\pi R^2}\frac{1}{r} \qquad (52)$$

where M is the total mass of the cloud and R is its maximum radius. For simplicity we choose a unit system with $G = M = R = 1$, again similar to previous authors using this test case. The particles are initially at rest and have an internal energy per unit mass of $u = 0.05\,GM/R$ and the specific heat ratio is $\gamma = 5/3$.

Once released, the system undergoes rapid collapse from inside out and reaches maximum compression at $t \approx 1.1$. Slightly before this time, enough kinetic energy has been converted into heat to build a pressure supported core in the center. Material that falls in later bounces off this core and is accelerated outward, forming a strong shock wave which interacts with the still infalling outer portion of the sphere.

The simulations presented below were run using the Gadget MAC (equation 41) for the tree with $\theta = 10^{-4}$. We used the spline kernel, equation 19, with a fixed length scale $h = 0.02$ for softening the gravitational potential of the particles. The hydrodynamic smoothing length was adaptive, so that each particle retains ≈ 50 neighboring particles (see section 3.2) at all times. The individual time step scheme described in section 2.4 was used for all runs. The parameters for the time step criteria (see equations 11, 12, 13, 36 and 37) were $\tau_{acc} = \tau_{vel} = \tau_{va} = 1$, $\tau_{\mathrm{CFL}} = 0.3$, $\tau_h = 0.15$. All runs were performed with the leapfrog integrator and some of them as well with the Runge-Kutta-Fehlberg integrator, with the latter using $\tau_{\mathrm{RKF}} = 10^{-5}$ (see equation 16). However, the difference between the two integration schemes is very small (see below), so we focus on the leapfrog simulations for most of the analysis presented here.

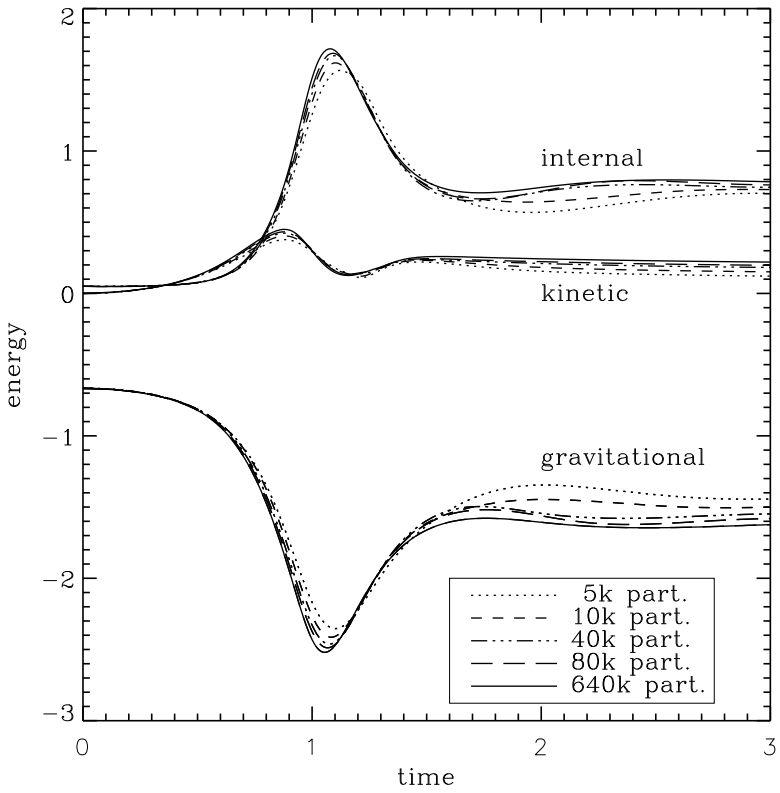

Fig. 2.— Time evolution of the total internal, kinetic and gravitational energy of the collapse test case. Resolution ranges from 5000 particles to 80000 particles, as shown in the box, with a run including 640000 particles run as a reference standard.

8.1.1. Fixed Artificial Viscosity

The evolution of potential, kinetic and internal energies of the system is shown in figure 2. The simulations have been performed at four different resolutions, using 5000, 10000, 40000 and 80000 particles. In addition, we have also run a simulation with 640000 particles as an internal reference standard against which the other runs can be compared. This will make it easier to assess the convergence of the simulations with increasing resolution and the effect of resolution on dissipative effects.

Comparing the 40000 and 80000 particle runs around the time of maximum compression ($0.7 < t < 1.5$), no energy component is different from the corresponding value in the other simulation by more than a few percent, with the largest differences coming near $t \sim 1.1$, when the gas is maximally compressed. Close examination of the curves show that slightly larger differences between these two simulations and the much higher resolution reference standard are present. For example, near maximum compression, the reference run has converted somewhat more gravitational energy into internal energy than any of the lower resolution realizations. We believe this occurs because the correspondingly lower dissipation at high resolution allows for higher infall velocities and correspondingly higher gas compression when the kinetic energy is converted into thermal energy. For the same reasons, the peak compression time occurs earlier as well.

During the equilibration and late stage adiabatic expansion phase of the system, after $t \approx 1.5$, differences between the 40000 and 80000 particle runs relative to the 640000 particle run become more pronounced. At progressively higher resolution, more energy remains in the form of internal

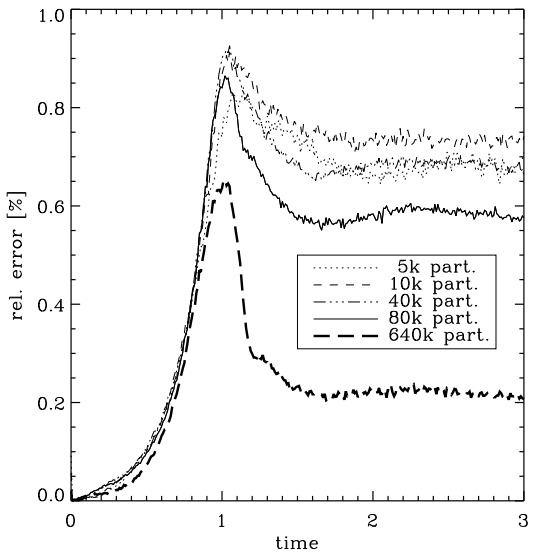

Fig. 3.— Time evolution of the relative error in total energy for the runs shown in figure 2.

energy, while the gravitational potential energy remains more negative, indicating that the gaseous core formed from the collapse is hotter and more tightly bound at higher resolution than lower. The largest deviations between the reference standard and either the 40000 or 80000 particle realizations occur between $1.5 \leq t \leq 2$, and thereafter decrease. Even at these times, the differences remain no larger than $\sim 5\%$, and we consider them to be well resolved, for the purposes required for this test.

At the lower resolutions of 5000 and 10000 particles, the internal and gravitational potential energies deviate from the 80000 particle run values by $\sim 10 - 15\%$ at maximum compression, with even larger systematic deviations at late times. These realizations have clearly not reached the level of convergence to the proper solution that is visible in the higher resolution runs.

The relative differences of a run integrated with the Runge-Kutta-Fehlberg integrator (not shown) as compared to a corresponding leapfrog run were never more than 0.63% in the internal energy, 0.44% in the kinetic energy and 0.40% in the gravitational energy. The corresponding difference in total energy reached a maximum of 0.32%. We do not consider these differences to be highly significant and so in the remainder of our discussions, we will present only the results of the simulations using the leapfrog scheme.

Figure 3 shows the relative error in total energy as a function of time, for each of the four realizations at differing resolution and our high resolution reference standard. In every case, the errors peak near $t \approx 1.1$, corresponding to the most compressed state of the system. Specifically, the maximum errors in total energy are 0.84%, 0.91%, 0.93% and 0.86% for the resolutions of 5000, 10000, 40000 and 80000 particles, respectively. Although we have observed error peaks both in these simulations and in those of other systems (see e.g. figures 10 and 11, below), their origin remains unclear. We have been unable to remove them completely through any combination of integrator or force accuracy settings, though their magnitude changes somewhat, while retaining computationally affordable simulations. Nevertheless, the overall magnitude of the error peak is small compared to ambiguities and shortcomings in the physical models of most systems of interest in astrophysical contexts, and we believe that the level of energy conservation produced by the code will be sufficient to model such systems accurately.

After peak compression, the errors fall to 0.7%, 0.74%, 0.67% and 0.57%, respectively, and remain nearly constant for the remainder of the evolution. The 640000 particle reference run reaches much lower errors: 0.65% at peak and 0.21% during the later phases of the simulation. Since both the integration accuracy parameters τ_i, $\tau_{\rm CFL}$, τ_h and $\tau_{\rm RKF}$ and the MAC setting for the gravitational force calculations were standard values and not particularly tuned for very accurate time integration, we believe this level of energy conservation is quite acceptable. Decreasing these parameters easily leads to energy conservation to $\sim 0.25\%$ or better throughout the simulation.

8.1.2. Time dependent Artificial Viscosity

We have also used this test case to study the differences due to use of the time dependent implementation of the artificial viscosity (AV), as defined in equation 29. In this test, we study simulations at a single resolution of 80000 particles and vary the implementation of artificial viscosity.

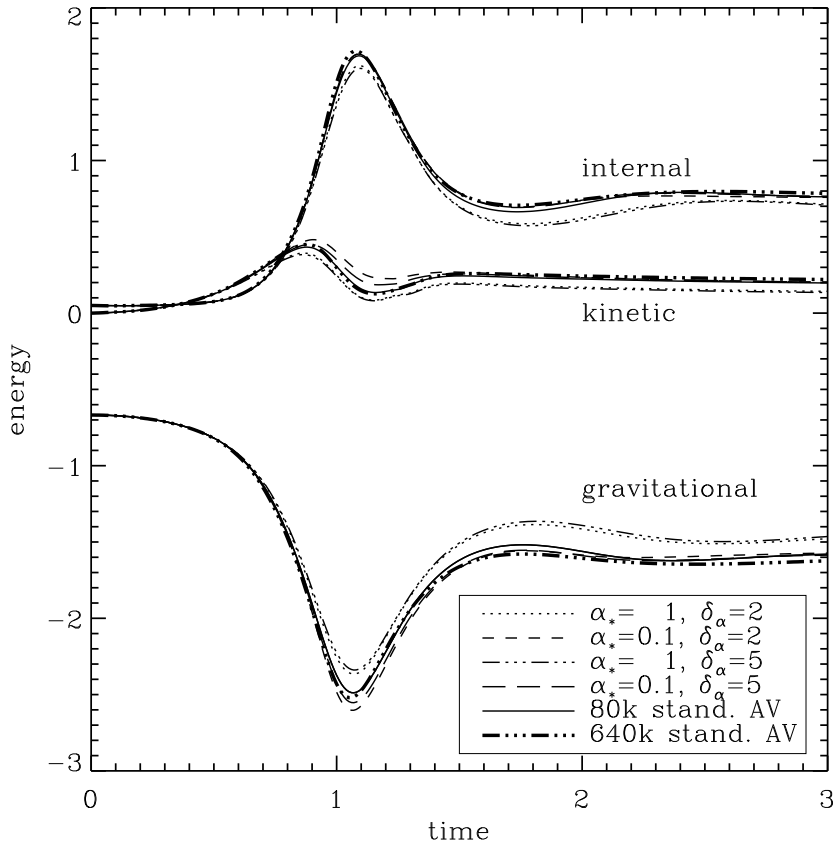

Fig. 4.— Time evolution of the total internal, kinetic and gravitational energy of the collapse test case for the model with 80000 particles, using different implementations for the artificial viscosity. In addition, a high resolution reference run with 640000 particles is plotted as thick dash-dotted line.

For reference and comparison to the results above, we also include the same 80000 particle simulation shown in figure 2, characterized by an AV formulation in its standard form (equation 26), with coefficients fixed at $\alpha = 1$ and $\beta = 2$, and the Balsara correction (equation 28). For comparison, we also include the high resolution reference simulation, with the same formulation. Models with the time dependent AV coefficients (equations 29–31) were run with different settings of the parameters α_* and δ_α, defining respectively, the value for the time dependent AV coefficient in quiescent flow and the scaled length over which that coefficient decays to its minimum value.

Figure 4 shows the time evolution of the total internal, kinetic and gravitational energy of the system for the 80000 particle model without time dependent AV and for four models at the same resolution identical except for changes in the time dependent viscous coefficients. The distribution of energies among components varies by as much as 10-20% for different settings, with the two $\alpha_* = 1$ variations producing the largest differences during and after peak compression. They are peculiar in the sense that substantially less internal and kinetic energies are generated, than in the fixed coefficient case. This is due to the higher dissipation in the initial infall phase ($t < 1$), which acts to slow the infall. The behavior appears qualitatively similar to that seen in the lower resolution variants in figure 2, for which we expect a correspondence between the lower resolution and higher dissipation. Since, for the parameters used, the viscous coefficient will always be larger than unity, the time dependent viscosity has the effect

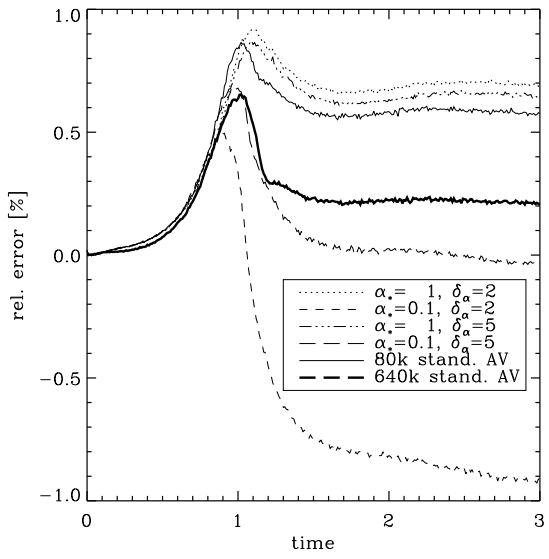

Fig. 5.— Time evolution of the relative error in total total energy for the runs shown in figure 4.

of lowering the effective resolution in the simulation, but since it is expected to be used only with a much smaller coefficient, the problem will not be severe in practice.

Both models with $\alpha_* = 0.1$ produce similar results for all three energy components and, more importantly, results which are very similar to the model with fixed AV. Only at and shortly after the maximum compression do differences appear, most visibly in the kinetic and gravitational energy components, where more resides in kinetic energy than in the fixed AV case, and less (more negative) in the gravitational potential energy. The differences from the 80000 run with fixed AV are at most a few percent over the entire duration of the simulation. More importantly, the models with $\alpha_* = 0.1$ resemble our high resolution reference run with 640000 particles better than the 80000 particle run with fixed AV. The effect of the time dependent AV, when used with values of $\alpha_* = 0.1$, is to simulate a higher resolution/lower viscosity system, and is clearly a very desirable effect. Of the two models with $\alpha_* = 0.1$, the one with $\delta_\alpha = 5$ is more similar to the 640000 particle reference run. At maximum compression, the minimum in the potential energy is not as deep as in the model with $\delta_\alpha = 2$, and the correspondence between the kinetic energies is closer as well. At late times, the model with $\delta_\alpha = 2$ tends to higher (less negative) potential energies, in fact higher than both the 640000 particle run and also higher than the 80000 run with fixed AV.

In figure 5, we show the relative error in total energy for the four models with time dependent AV, the fixed AV reference model and the high resolution reference run. The high α_* models have larger errors than the model with fixed AV, with the $\delta_\alpha = 5$ model coming slightly closer to the fixed AV result than the $\delta_\alpha = 2$ model. The two low α_* models develop much smaller errors during maximum compression, but after the peak, their energy errors go in the opposite direction. At the end of the simulation, the error of the $\alpha_* = 0.1$, $\delta_\alpha = 2$ model reaches 0.9%, as large or larger than the peak errors of both the fixed high α_* models, and is still growing. The model with $\alpha_* = 0.1$, $\delta_\alpha = 5$ shows the best performance overall on this problem. Interestingly, its energy error decreases after maximum compression, similar to the corresponding $\delta_\alpha = 2$ model, but becomes almost flat and is 0.04% at the end of the simulation. At and around the peak, its behavior is very similar to the high resolution reference run. After $t \approx 1.2$, it falls to lower errors than the high resolution run, but overall it is still the model with comes closest to the 640000 particle reference run.

The differences seen in figures 4 and 5 are due to the differences in the ability of one set of AV parameters to model shocks and compressions with more fidelity than another. Because modelling such shocks and compressions is very sensitive to how AV is implemented in a code, we move now to a closer examination of the radial density profiles of the simulations, in which we expect a shock front to be clearly visible.

Figure 6 shows profiles at four times during the evolution for each of the models, and figure 7 shows a close up view of the shock front at $t = 0.9$. For comparison, each plot the result of a one dimensional simulation of the same system using the Piecewise Parabolic Method (PPM) with 350 zones, taken from Steinmetz & Müller (1993). We expect better AV behavior in the SPH runs will be reflected in curves more similar to that obtained from the PPM run. As an additional point of reference, we also show the result of the 640000 particle run in the close up view of the shock front,

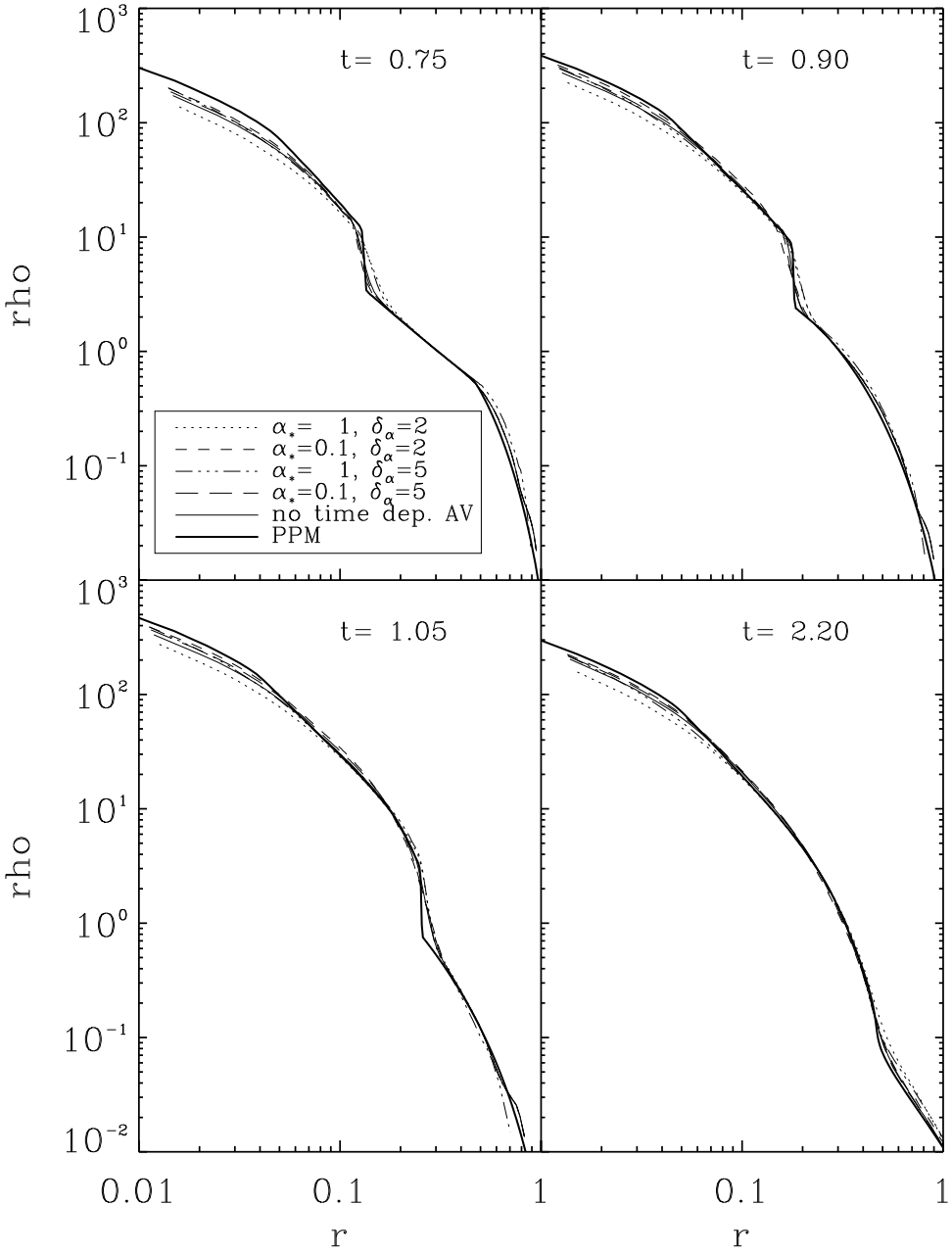

Fig. 6.— Radial density profile of the models shown in figure 4. The thick solid line represents results of a PPM simulation by Steinmetz & Müller (1993). The upper two panels represent points in time before maximum compression in the center, the lower left shows the system at maximum compression and the lower right is after the shock wave has passed through most of the system.

shown in figure 7.

In every realization, the overall density profiles are similar, with high densities in the center, a discontinuity further out, and a decrease to low

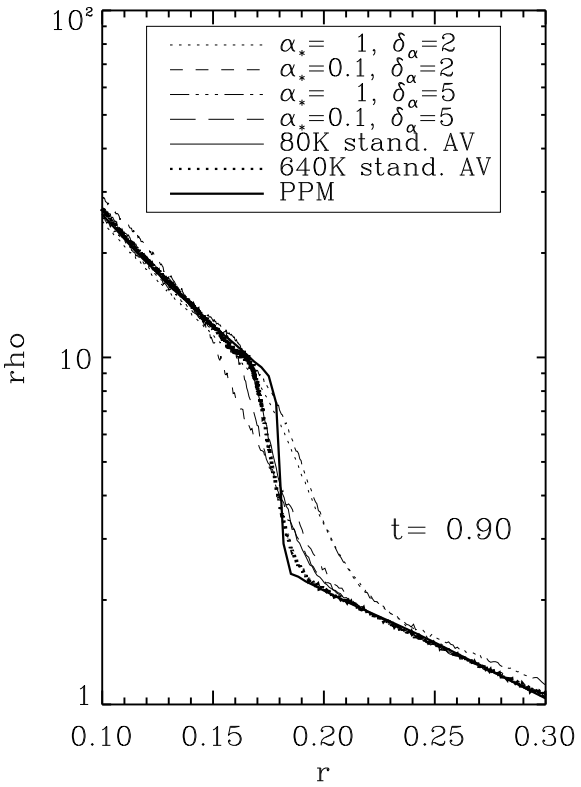

Fig. 7.— Radial density profile of the shock wave at $t = 0.9$. The thick solid line represents results of a PPM simulation by Steinmetz & Müller (1993).

densities at the largest radii. The similarities give confidence that the gross behavior of all variants produce sensible results. There are differences specific to each realization however. For example, the high viscosity models disagree most at the largest radii, where the densities are overestimated relative to the PPM model, especially at late times when the material is expanding freely. The low viscosity and fixed AV models agree well with the reference PPM curve there at all times. At small radii, all the SPH models underestimate the central density. The low viscosity models ($\alpha_* = 0.1$) agree best, followed by the fixed AV and long decay constant, high viscosity model and, worst of all, the model with $\alpha_* = 1$ and $\delta_\alpha = 2$, for which a peak density nearly a factor two below that of the PPM run is observed. The central densities agree with the PPM realization to within $\sim 3.7\%$ in the 640000 particle reference run at $t = 0.75$ (though are not plotted in figure 6, in order to reduce clutter) and, at late times, this SPH run actually overestimates the central density by as much as 10%. Using the central density as a metric, the models with 80000 particles used for this AV test are not fully converged, but the similarity between the PPM and SPH methods at higher resolution indicate that they do converge to very similar results, given sufficient resolution.

During the compression phase, the high viscosity models show significant broadening of the shock front, as is expected, but also an outwards position shift of the foot of the shock with respect to both the PPM results and the other SPH models, especially visible in the $t = 0.9$ close up view in figure 7. The low viscosity models reproduce the shock front more accurately and at the same radial position as the PPM model. In front of the shock front (radially outwards), both low viscosity models are very similar to each other and to the fixed AV model, while behind it, the $\alpha_* = 0.1$, $\delta_\alpha = 2$ model exhibits a much shallower rise than it should. The longer decay constant variant ($\alpha_* = 0.1$, $\delta_\alpha = 5$) shows a much steeper profile there, but still not as steep as the fixed viscosity version, which appears to be the best reproduction of the shock front. The high resolution model shows closest correspondence to the PPM density curve at the foot of the shock, but the high density side nearly over lies its lower resolution cousin, also with fixed AV. Later, near maximum compression ($t = 1.05$–lower left panel of figure 6), simulations with all of the AV implementations misplace the shock front, to slightly larger radii.

Given the behaviors we see in the shock structure and in both the components of the energy and the conservation of total energy, we conclude that the time dependent AV formulation allows a simulation to be performed with dissipation from artificial viscosity at a level comparable to that of a much higher resolution simulation that does not include such a formulation. Our results show that the parameters $\alpha_* = 0.1$, $\delta_\alpha = 5$ should be used for best results. Simulated with these parameters, our test system yielded the best overall conservation of energy, closest correspondence of all energy components to a high resolution, fixed AV reference model, at the cost of a slightly more broadened shock front, compared to a run with fixed AV. Otherwise the shock capturing abilities are comparable to the standard formulation with

fixed AV. In the absence of shocks, the time dependent AV formulation yields better results than the time independent formulation. Hence we favor the use of the time dependent formulation.

These settings differ from those recommended by Morris & Monaghan (1997), of $\delta_\alpha \sim 2$ for the decay length. Such differences illustrate the problem dependent nature of the settings themselves, and are most likely consequences of the presence of shocks of different strength in different problems. Users of VINE who wish to employ the time dependent AV formulation may need to account for such differences when choosing appropriate viscous parameters.

8.2. Elliptical-Elliptical Merger Tests

The quality of hydrodynamic simulations with VINE has been studied in detail in section 8.1. In order to assess VINE's ability to correctly evolve N-body models, we compare the evolutionary trajectory of models simulated with VINE and with Gadget-2. We use a merger simulation of two elliptical galaxies (a 'dry merger', see e.g. Naab et al. 2006a) as a test problem. As the gas fraction in such systems is very low, a pure N-body representation of the galaxies, or one including only a small fraction of gas particles, is a realistic model. The evolution of such a dry merger requires correct behavior of the code over a wide range of dynamical timescales. While the merger event itself is short compared to the duration of disk galaxy mergers, its correct simulation requires accurate time integration of particle trajectories in a highly time dependent gravitational potential, requiring a wealth of interesting dynamical behaviors to be accurately modeled. The correct simulation of such a system is therefore a challenging test case.

The initial conditions for our tests are created by setting two elliptical galaxies on a parabolic orbit, which is a reasonably realistic setup for the orbit in a cosmological context (Khochfar & Burkert 2006). Each elliptical galaxy is understood to be the remnant of a prior disk merger event whose components had a 1:1 mass ratio and in which each of the spiral galaxies consisted of a stellar disk and bulge as well as a dark matter halo. We use three somewhat different variants of this overall setup, in order to allow us to derive parameters for both codes that are reasonable for a wider range of merger applications and are not fine tuned for just one particular problem. The different mass ratios and matter distributions in the galaxies provide some confidence in this respect. We demonstrate below that our choice of parameters, although derived from a different merger morphology, still provides comparable energy conservation for our elliptical merger test case and also yields very good agreement between the two codes regarding the properties of the remnant galaxies.

8.2.1. Requirements for sensible code to code comparisons

For any physical model sufficiently complex to be interesting for anything more substantial than an academic exercise, the only option for validating the numerical code used to simulate it is to compare against results obtained from other codes. In this sense, validation effectively assumes that while those other codes have been developed to solve the same set of equations, they have been constructed differently enough, and have been validated independently based on a sufficiently different set of criteria, to ensure that they are effectively neutral standards against which the results of another code can be measured. The question of whether or not this assumption itself can be validated is unlikely ever to be answered satisfactorily. Nevertheless, we will proceed as if it has been, and compare the results of a simulation run using VINE with the results of the simulation using the same initial conditions using the publicly available and widely used code Gadget-2 (Springel 2005).

Of course, various input settings used for a particular simulation will directly affect both its accuracy and the overall results that are obtained. It is trivial to run one or both codes with inappropriate parameters for the given problem and thus find disagreement in the results. Our first step in making comparisons must therefore be to minimize the differences originating in different settings by running an extensive set of tests prior to the actual comparison of the elliptical merger simulations that are our main interest. Only when this process is complete can we attempt to characterize results obtained from each code on an even handed basis.

For our comparisons, VINE uses its leapfrog integrator (section 2.2) with individual timesteps (section 2.4) and the Gadget MAC (equation 41).

The Gadget-2 code uses similar, but not identical, schemes for each of these three choices. With these choices given, there still remains a very large number of additional degrees of freedom defining all of the different code settings available to the user, which are both different in each code and, to the extent that a correspondence exists, may require different values to produce results of similar accuracy. It serves little purpose to match the settings of every one of these parameters, especially if the normal settings for one or the other code are significantly different. Instead, we choose to match the results of the codes to what we believe are among the two most crucial parameters: first the gravitational force accuracy, as controlled through the MAC parameter θ (equation 41), and second, the tolerance parameters, τ, used in the integration scheme (equations 11–13).

Even with only two parameters to adjust, some differences are unavoidable. For gravity, VINE uses a multipole series truncated at quadrupole order and the hexadecapole based Gadget MAC described in Springel et al. (2001), while Gadget-2 uses a monopole truncated series and a quadrupole based form of the MAC. Each code will therefore require different values of the MAC parameter θ to obtain similar force accuracies. When the mass distributions within a given tree node are very inhomogeneous, as will certainly be the case in the intermediate stage of this simulation, truncation errors in the multipole summation will be systematically larger than for other distributions, resulting in larger errors for a given setting. In addition to the MAC differences, both codes test whether or not a particle and node overlap in space, but use different procedures to define approximate sizes for the nodes. While Gadget-2 errs conservatively by requiring the particle to lie outside of a box which is $\sim$20% larger than the actual node (see equation 19 in Springel 2005), VINE uses a conservative definition of the node size itself (see Vine2), which also errs by over-estimating the node size. Which variant yields the more efficient criterion is not clear.

Looking at the time step criteria, Gadget-2 does not use timestep criteria that include the velocity (equations 12–13). Also the actual equation implemented for the force criterion, equation 11, included the tolerance parameter under the square root. So we expect to find similar integration accuracy for both codes at different values for the accuracy parameters. For simplicity in the discussion below, we will denote the accuracy parameters with τ_{acc}, but actually include all three τ values in equations 11–13 in VINE (set to the same value). We will use the same symbol, τ_{acc}, for Gadget-2 to mean the single active accuracy τ parameter for Gadget-2, despite the different functional form of its implementation.

8.2.2. Determining comparable integrator and tree opening parameters for each code

In order to find corresponding θ and τ parameters for the two codes, we use lower resolution merger setups than in our main comparison. Otherwise, repeating each test many times as we constrain the acceptable parameter range would require prohibitively large amounts of computing time, particularly for tests to determine τ_{acc}.

We constrain the values of the parameters using a two step process. First we determine node opening parameters which yield similar force errors for the particles. For both codes, we require that less than 1% of particles have gravitational force errors $\geq 1\%$ and that 99% of the particles have force errors $\leq 0.2\%$. We use two limits in order to ensure both that the full distribution of errors is low and also that a small tail of the distribution with high errors does not adversely affect the simulation outcome. For the second step, we use these MAC values in several full simulations with each code, in which we vary τ_{acc} and we require that the error in total energy never exceed 1% at any time during the simulation. While any particular choice of error limits is somewhat arbitrary, some choice must be made. These values reflect those used most frequently in our production runs.

The initial condition used for the tests to constrain the MAC setting is a disk merger simulation with a mass ratio of $3:1$, in which the more massive galaxy is modelled with 60000 particles for the stellar disk, 20000 for the stellar bulge, 20000 for the gaseous disk component and 120000 for the dark matter halo. The disc has a gas fraction of 10% and the bulge contains one third of the total mass of the disc. The less massive galaxy contained the same components, modelled with one third as many particles in each. The galaxies are on a parabolic orbit. For more details of the setup, we refer the reader to the merger se-

tups used in Naab & Burkert (2003) and Wetzstein et al. (2007).

The setup was first simulated with VINE at very high accuracy. From this reference run we selected snapshots at three different times corresponding to the initial condition, just prior to the merger so that one galaxy is located immediately adjacent to the other, and a very late stage, well after the strongest interactions between the galaxies are complete. We will refer to these three stages as the initial, intermediate and late stages, respectively. For the VINE calculations, accelerations for each snapshot were calculated directly from the code, with the results saved to files for later analysis. In order to compare identical mass distributions with the two codes, the three snapshots of the reference run with VINE were converted to a data format readable by Gadget-2. Each converted snapshot was read in to the Gadget-2 code which calculated the gravitational forces for the particle distribution, again saving the results for later analysis.

For all three stages, we investigated the relative error in gravitational forces as a function of the tolerance parameter θ used in the relevant MAC. Our procedure of comparing three different stages of the simulation still does not necessarily guarantee that the force accuracy over the entire simulation will remain comparable, but we have some confidence that the deviations will not be too large because we test three very different mass distributions, representative of the range of mass distributions covered by the simulation.

Figure 8 shows the error magnitudes for the median, 95% and 99% error limits for each code at three different times in the simulations. As expected from the differences in the MAC definitions, comparison of the error limits for the same value of the opening criterion show that the limits for VINE are not the same as those for Gadget-2, with the latter being considerably larger. The differences are reflected in the required opening parameters: in order to fulfill the error limits described above, we require a value of $\theta = 5 \times 10^{-3}$ for VINE and $\theta = 2.5 \times 10^{-4}$ for Gadget-2. The parameter value required for Gadget-2 lies well below that at which the intermediate time error curve diverges from the curves defining the intermediate and late time error distributions. Its value is therefore not determined by any peculiarities of the mass distribution which caused the divergence in the first place and we do not expect that its value will be significantly altered in other morphologies.

Beyond the simple statement of the MAC parameters required for each code, it is of some interest to examine figure 8 further, to study the differences between the codes and, hopefully, to better understand the consequences various choices of the criteria may have on the simulations.

For VINE, the error magnitudes for all three limits follow similar patterns as functions of the opening parameter, at all three times. The error distribution at the initial time lies somewhat above those of the intermediate and late time distributions, presumably due to the smoother particle distribution present at that time compared with that at later times. Even so, its error distribution at the most permissive opening criterion for which we obtained data terminates well below the 1% level. Also, the error distribution curves appear to decrease their slope at larger $\log\theta$ values, rather than continuing a linear increase. Detailed tests discussed in Vine2 demonstrate that both phenomena continue towards still larger θ values and that the error limits never increase beyond $\sim 2-3\%$, even in the limit of a uniform density (where we expect small force magnitudes due to the greater level of cancelation of partial forces), a fact that will be very beneficial in general for most simulations.

For Gadget-2, the error distributions of the initial and late time particle distributions lie virtually on top of each other. The intermediate time distribution deviates quite significantly, exhibiting a substantial population of particles with force errors approaching 10%–a level clearly unacceptable for most simulations of astrophysical interest and as much as a factor of several larger than either the initial or late time distributions. Also, the error limits for a given opening criterion do not appear to reach any limiting values as θ increase towards more permissive limits. These observations are important because they demonstrate that the node opening criterion must be chosen with great care in Gadget-2, if large errors are to be avoided.

With the MAC parameters defined, we turn now to runs of a set of merger simulations in which we varied the integration accuracy τ_{acc}. For this test, we choose a lower resolution than in the de-

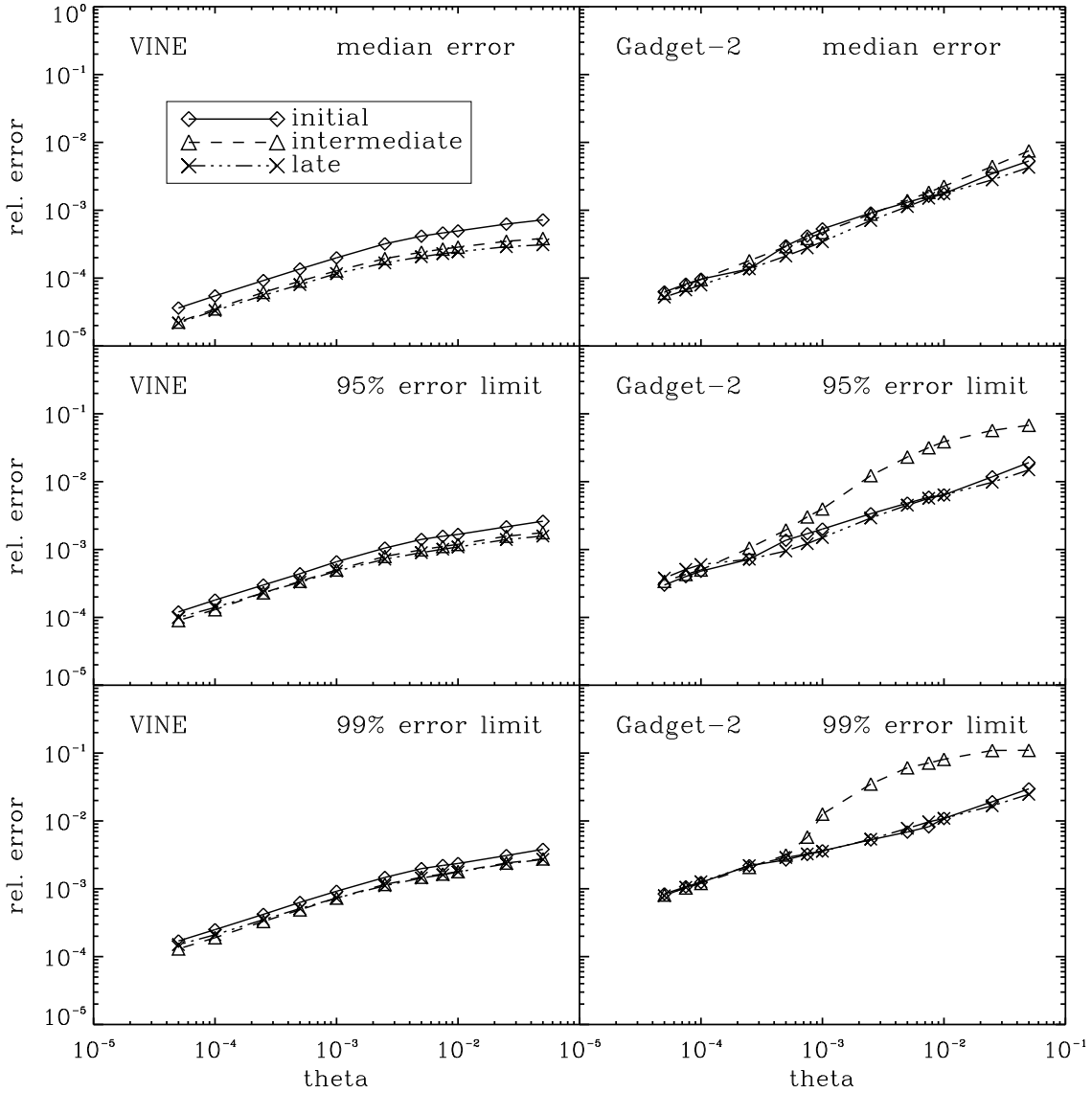

Fig. 8.— Relative errors of the gravitational forces as a function of tree node-opening criterion θ for VINE and Gadget-2. From top to bottom, the median, 95% and 99% error values taken over all particles in each simulation.

termination of the force tolerance above, as several full simulations with each code are required to find reasonable accuracy parameters. In addition, to further our goal of finding parameters which are not specifically tuned to a particular merger setup, we use a different setup for these merger simulations. Our criterion for sufficient integration accuracy is to impose an upper limit on the error in total energy, therefore we must define a model with *only* N-body particles, rather than including particles to model a gas component as well. Models which include gas require some mechanism to cool the gas during the encounter in order to balance the expected heating due to shocks. Whether the gas is allowed to cool radiatively or is modeled with an isothermal equation of state (assuming a balance between shock heating and radiative cooling), the total energy of the system is not con-

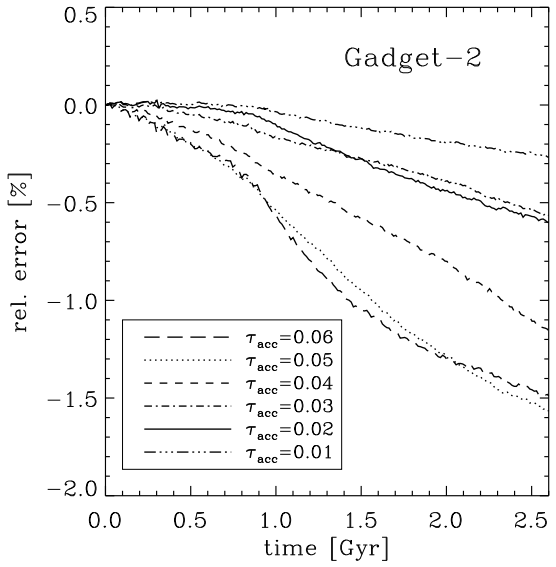

Fig. 9.— Relative error in total energy for the low resolution merger models, using various integration accuracy parameters τ_{acc} with Gadget-2.

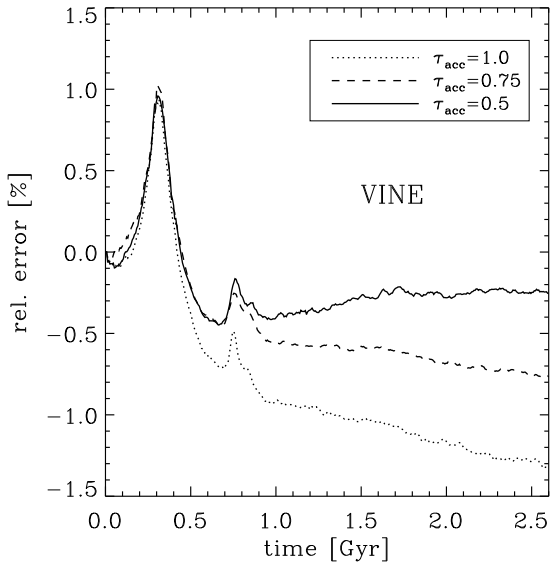

Fig. 10.— Relative error in total energy for the low resolution merger models, using various integration accuracy parameters τ_{acc} with VINE

served. In contrast, limiting the model to N-body particles allows the model to retain (in theory at least) the energy conservation condition required to constrain our accuracy parameters.

We use an equal mass merger in which each galaxy as modeled with 10000 particles for the stellar disc, 2000 for the bulge and 20000 for the dark matter halo. Again, we place the galaxies on a parabolic orbit. We evolved this setup with both VINE and Gadget-2, adopting the force accuracy parameters θ from above and using different values for the integration accuracy parameter τ_acc. We first verify that with this choice for θ, the force errors during the first close encounter, for which the errors become largest, satisfy the same force error limits defined above. Both the median error and the 99% error limit are very similar to those found in the more detailed study, providing some confidence that our choice of parameters is indeed suitable for a wider range of merger simulations.

In figures 9 and 10 we show the error in total energy for τ_{acc} ranging from 0.01 to 0.06 for Gadget-2 and from 0.5 to 1.0 for VINE, respectively. In all of the Gadget-2 simulations, the curves exhibit a slow secular drift towards energy loss from the system, with the magnitude of the loss being larger for the more permissive integrator settings. Interestingly, the pair of simulations with $\tau_{acc} = 0.05$ and 0.06 yield curves quite similar to the other, with near identical magnitudes throughout and only small relative variations as one curve increasing or decreasing above or below the other at different times in the simulation. A similar phenomenon is present in the $\tau_{acc} = 0.02$ and 0.03 pair, but with smaller magnitudes. Since for the latter pair, the error magnitude of the $\tau_{acc} = 0.02$ yields better behavior during the close encounters, we choose this choice for the Gadget-2 code.

The overall shape of the energy error curves for the VINE simulations are quite different from those for the Gadget-2 runs. Instead of a secular drift, the errors peak at $\sim$1% above their initial values at the first close encounter of the two galaxies, then decrease to values below the initial value as they move apart again. After a second, smaller peak at the second encounter, the curves for the most permissive integrator settings slowly decrease further. The peak error magnitude appears quite insensitive to the integrator setting, with all three choices yielding nearly the same result. As noted briefly in our hydrodynamic test problem above, we know of no specific feature either in the integrator itself, or in the calculations of the gravitational forces, which would

lead to such behavior. After the main merger events, the model with $\tau_{acc} = 0.5$ shows only very small evolution of the error. The least accurate model, $\tau_{acc} = 1.0$, clearly fails to meet our limit of 1%. The one with $\tau_{acc} = 0.75$ reaches errors slightly above 1% at the peak, while the one with $\tau_{acc} = 0.5$ has less than 1% error at the first encounter. Afterwards, its evolves with much smaller errors than the other two. We therefore we adopt an integration accuracy parameter of $\tau_{acc} = 0.5$ for VINE.

8.2.3. Results of full scale merger simulations run with VINE and Gadget-2

Using the above choices for the accuracy parameters θ and τ, we ran a series of full scale elliptical merger models for 3.9 Gyr with both VINE and Gadget-2. As above, the initial condition for the tests is created by setting two elliptical galaxies on a parabolic orbit. In these tests, the stellar disk, bulge and a dark matter halo were modelled with 60000, 20000 and 120000 particles, respectively. Therefore, each elliptical galaxy in our test simulation consists of 160000 stellar particles and 240000 dark matter particles so that the entire simulation contains 800000 particles. For more details, we refer the reader to Naab et al. (2006a).

Figure 11 shows the evolution of the error in total energy for both codes. The error behaviors of the two codes correspond nearly identically to that seen in section 8.2.2 above, but with smaller magnitudes, as expected from the higher resolution models in this test. The maximum error in the VINE simulation reaches $\sim 0.48\%$ at the time of the first close encounter, while in the Gadget-2 simulation, it reaches 0.44%, at the end of the simulation. Between and after the merger encounters, the energy error in VINE stays roughly the constant, particularly at late times (after $t \approx 0.5$, while for Gadget-2, the error continues to increase until the end of the simulation, though slowly.

A direct examination of the overall morphologies of the mergers at different times, and as realized by the two codes, does not prove to be of great use to compare differences in the evolution provided by one code over the other. The features of the merger as evolved using one are present in the other, and we are not able distinguish between them in any quantitative fashion. Instead, and in order to compare the results of both codes more quantitatively, we explore the behavior of the centers of mass of the particles making up the stellar

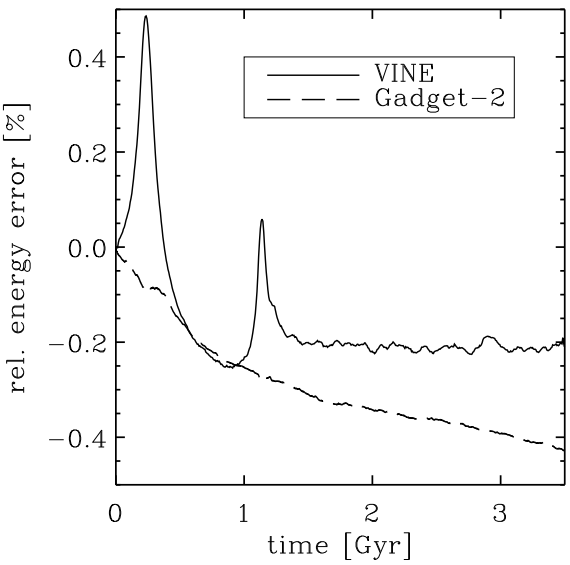

Fig. 11.— Comparison of the relative error in total energy as a function of time for the elliptical merger simulation.

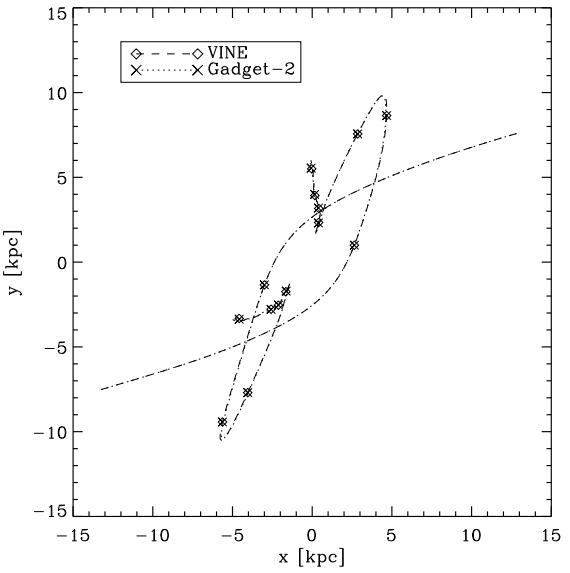

Fig. 12.— Center of mass trajectories in the orbital plane for the stellar component in each galaxy. The symbols along the trajectories represent the following times in Gyr: 0.25, 0.5, 1.0, 1.25, 1.5, 2.0, 3.5

component of each galaxy. The stellar component, as the most luminous, is most interesting to compare because it resides in the central part of the galaxy and dominates the potential there. Tracking its behavior therefore means that we are tracking the dominant component of the most dynamic region during the merger.

Figure 12 shows the center of mass trajectories of the stellar component of both galaxies, as realized by each code. The trajectories of the galaxies simulated with VINE and Gadget-2 lie essentially on top of each other at all times. We have also plotted symbols along the trajectories to indicate different epochs in the evolution which have been chosen to sample the trajectories over a wide range of evolutionary stages. The position of each snapshot along the trajectories also matches between the Gadget-2 and the VINE simulation.

In each case, the two galaxies have a first close encounter after which they move to larger distances again. Then they reach a maximum separation, turn around and approach each other again and finally merge. During and after the final merger, the center of mass trajectories of the stellar components do not converge to a single point, even though the galaxies as a whole merge. Instead, they move apart as a consequence of the asymmetric distribution and trajectories of tidal debris well outside the newly formed elliptical galaxy (some $\approx 15\%$ of the total stellar mass) which was created both during the merger simulation and assumed in the progenitor ellipticals, themselves taken to be remnants of mergers. The tidal debris is accelerated outward during the merger event and thus the center of mass of the star particles of a progenitor galaxy moves gradually away from the center of the new elliptical, as the tidal material moves to increasingly larger distances. The center of mass of the system as a whole is of course not affected by this behavior.

We conclude our comparisons of the Gadget-2 and VINE simulations by comparing the merger remnant produce by each simulation. In figure 13 we plot the rotation curve of the galaxy and its surface density profile. The circular velocity profile of the simulations lie nearly on top of each other for both stars and dark matter, as do the surface density profiles over nearly their entire range. Only in the outer regions of the galaxy, where resolution begins to degrade, do very small differences become visible. Thus the mass distributions of both the luminous and the dark component of the merger remnant agree well between the two codes.

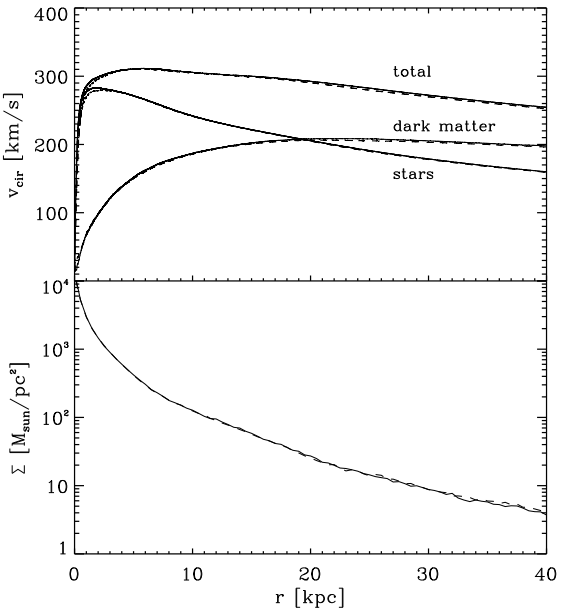

Fig. 13.— Rotation curve (upper panel) and surface density profile (lower panel) of an elliptical-elliptical merger. The solid lines are results using VINE for the simulation, the dashed lines show Gadget-2 results.

Given the complexity of the problem and the very different features and algorithms implemented in the two codes, we conclude that both codes agree very well on this demanding N-body test case.

9. Performance of the Code

In Vine2 we present detailed timings of the code on both serial and parallel test cases. Various optimizations to the code are described and their effects on the performance of the code investigated. We will therefore leave our most detailed discussions of the code's performance for Vine2, and perform only speed comparisons between VINE and Gadget-2, using the same simulation used for the accuracy comparisons in section 8.2.3. The performance comparison to Gadget-2 will also serve as a frame of reference for the timings presented in Vine2.

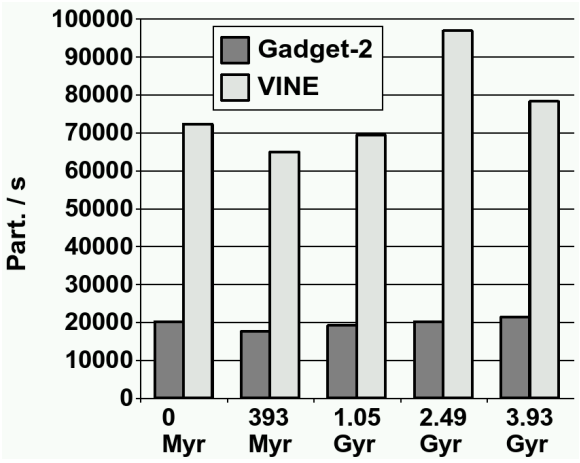

Fig. 14.— Calculation rate in particles per second for the gravitational force calculation of all 800000 particles in an elliptical-elliptical merger simulation. The results are shown for different stages of the simulation. Note that the force accuracy of VINE in this test is slightly higher than that of Gadget-2 and VINE calculates force and potential, while Gadget-2 only calculates the force. For more details, see text.

9.1. Speed comparison of the gravity calculation in VINE and Gadget-2

For this analysis we use snapshots of the simulations output by each code at five different times of the simulation, to account for variations in the speed due to the different mass distributions, seen at different times. We adopt the same node opening criteria in section 8.2.2, which yielded the same force accuracy for each. Specifically, we used parameters of $\theta = 5 \times 10^{-3}$ for VINE and $\theta = 2.5 \times 10^{-4}$ for Gadget-2, which guarantee force errors of less than 0.2% for 99% of the particles. As before, we convert an output dump from the VINE version of the simulation to Gadget-2 format, and use that particle distribution in the rate calculation, in order to eliminate ambiguities due to differences in particle distribution in the calculations of the rates.

Figure 14 shows the speed for the gravitational force calculation on all particles using these accuracy parameters, using 8 processors of the SGI Altix at the University Observatory, Munich, i.e. 8 Itanium 2 processors running at 1.5 GHz. The snapshots at $t = 0$ is the initial setup, where the two galaxies are still well separated. At $t = 196$ Myr, the galaxies have their first close encounter. They pass by each other and their distance increases until the turnaround point. The snapshot at $t = 525$ Myr is shortly before this point. Then they two galaxies merge and form a new elliptical. The snapshot at $t = 1.25$ Gyr is after the merger but before the system has had time to relax completely. The final snapshot st $t = 1.97$ Gyr is at a very late stage when the newly formed system has already largely relaxed. For the 5 snapshots from $t = 0$ to $t = 1.97$ Gyr the speed of the gravitational force calculation using VINE ranged between $\sim$ 65000 and $\sim$ 95000 particles per second, while that using Gadget-2 ranged over values slightly less than 20000 particles per second, yielding speedups of a factor between 3.6 and 4.8 for VINE over Gadget-2.

Before concluding that VINE's gravity calculation is indeed faster than Gadget-2's, we must be certain that neither code settings, nor the details of the calculations themselves have not unfairly biased the comparison. We recall here again, that although we have attempted to use code settings that yield similar error limits for both codes at all times, differences between the resulting error limits will inevitably remain, both between the two codes and for the same code at different times. What affect do these variations have on the derived rates?

For example, for the node opening parameter settings used in our test, figure 8 shows that the error limits for the Gadget-2 runs are nearly identical at all three times but for VINE, different stages of the simulation clearly exhibit different error magnitudes. The intermediate and late stages of the simulation have lower errors than what we impose for the whole simulation. In principle, they would allow a larger parameter setting at those times if desired, increasing the calculation rate. Since the error magnitudes for the Gadget-2 runs are the same at all times, no similar tuning would be possible there. Further, simply relaxing our error requirement itself does not appear advisable, at least for this problem, because of the intermediate time curve's divergence from the other two curves, towards very much larger errors.

An additional difference is that VINE always computes both the gravitational potential and the force, while Gadget-2 computes only the force.

While it is possible for Gadget-2 to compute the potential as well, it requires a second tree traversal and computation, effectively doubling the time for the combined calculation. Such a fundamental difference between the algorithms in the codes would unfairly distort the comparisons between the timings in the two codes, so we compare the combined force and potential calculation of VINE with the force only calculation of Gadget-2.

Taken together, each of these restrictions on the performance measurements should result in a slight bias in favor of Gadget-2, making the actual result favoring VINE even more significant. We conclude that the calculation of the gravitational forces in our test case is considerably faster with VINE than with Gadget-2. The speed differences between VINE and Gadget-2 also provide some confidence that full simulations are also faster with VINE than with Gadget-2, as the gravitational force calculation which has been timed here typically consumes some 70% to 80% of the total time required for a simulation. It is, however, extremely difficult to make fair comparisons of the overall speed of two codes for a full simulation, and we have therefore not attempted to do so here. The actual speed differences between codes will likely vary somewhat between different problems. We have little confidence that any node opening criteria, or any integrator accuracy parameters set to provide correspondence at one time, for one problem, will result in similar enough time step distributions over an entire simulation to allow a fair or sensible comparison to be made. In spite of these concerns, the results in hand demonstrate that VINE will be an excellent choice for use in efficiently solving most problems encountered in astrophysical contexts.

10. SUMMARY

In this paper we have introduced VINE, a hybrid N-body/SPH code which uses a binary tree structure for the calculation of gravitational forces. It is a very modular and flexible code which allows the user to compile and use only those modules which are required for simulating the physics of the problem at hand. This modular structure also makes it fairly easy to exchange such modules for new ones if needed or to add others to implement new physics or numerical features.

The code includes both a Runge-Kutta-Fehlberg and a leapfrog integrator, which can be chosen by the user at compile time. Both can make use of an individual particle timestep scheme. We have described the SPH implementation in VINE, including details of the symmetrization and the scheme for adapting the smoothing lengths. VINE includes the capability for calculating gravitational or other long range forces using a tree structure to organize and sort particles into near and far interactions, and an outline of the techniques are described here. We describe the implementation of periodic boundary conditions used in both the gravity and the SPH part of the code. Finally, we demonstrate the capability of the code to accurately simulate both hydrodynamic and N-body problems using, in the first case, the collapse of a gas sphere as a test problem and, in the second, an elliptical-elliptical galaxy merger.

We have demonstrated that the code performs well on a standard, but somewhat contrived, test problem with a well known result. More importantly, we demonstrate that it performs well on a full astrophysical simulation of the merger of two elliptical galaxies, in comparison to the publically available Gadget-2 code. The performance of the gravitational force calculation in this test, the most costly component of simulations including self-gravity, was far superior to that of Gadget-2, ranging between 3.5-4.8 times faster, at different times in the calculation.

10.1. Additional optimizations, features and future directions

VINE, as it has been presented here and in Vine2, will be released to the public under GNU Public License. We hope that it will become a useful tool for use on a wide variety of problems. Due both to the features included in its base form and the overall modular structure of its design, we expect that it will not be overly burdensome to adapt the code for use on other problems, or to add additional features to it.

At present, the code exists in a flexible, 'base' version, which includes a number of basic physics packages common in many astrophysical contexts, but is by no means complete. We expect that other workers will wish to incorporate packages not included in the base version, or to make changes to it, in order to advance their own re-

search goals. Already, we foresee additions that are required to satisfy our own research goals, for example. At the University Observatory in Munich, efforts are underway to develop modules to implement radiative cooling, subgrid models for star formation on galactic scales, stellar feedback, black hole accretion, AGN feedback, inflow boundaries and ionizing radiation. These packages are at many different stages of development, with some essentially complete, and others merely in planning stages. VINE has also been the basis for an implementation of the SPH method on special purpose, reconfigurable hardware, so called FPGA (*F*ield *P*rogrammable *G*ate *A*rray) boards (Wetzstein et al, in preparation).

In current form, VINE's SPH module does not include strength and damage models needed to simulation systems including solid bodies. In future work, we intend to integrate into VINE such a model, present in a much earlier cousin (Benz & Asphaug 1995), for use in modern simulations of, e.g., giant impacts between planetary bodies or of asteroids on earth. These types of calculations may also require different or more advanced treatments of the hydrodynamics, such as are available using the Moving Least Squares Particle Hydrodynamics (MLSPH) approach of Dilts (1999, 2000), which may be substituted for VINE's standard SPH module.

Other features of interest include alternate integrators. The VINE framework could, for example, be adapted to accommodate integrators specially designed for use in highly accurate N-body simulations (Aarseth 1999; Levison & Duncan 1994). A KDK variant of the leapfrog integrator (Springel 2005) is also of great interest, due to its stability properties and the fact that force calculations can be parallelized with higher efficiency when used in individual time step mode.

We expect that if VINE finds wide use in the astrophysical community, many other modules may be developed, beyond those suggested here. We hope that the code will be as useful to others in the astrophysical community as it has been for us so far.

10.2. Availability of the code

The code is available to the public under GNU General Public License version 2 from the authors or via download at the USM website: http://www.usm.lmu.de and at the UKAFF website: http://www.ukaff.ac.uk.

We wish to thank Willy Benz for his generous gift to so many, over so many years, of his SPH wisdom and the original code on which VINE is based. Some of the computations reported here were performed on the SGI Altix 3700 Bx2 supercomputer at the University Observatory, Munich, which was partly funded and is supported by the DFG cluster of excellence "Origin and Structure of the Universe" (www.universe-cluster.de). Other computations and most of the code development used facilities at the UK Astrophysical Fluids Facility (UKAFF). Portions of this work were carried out under the auspices of the National Nuclear Security Administration of the U.S. Department of Energy at Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396, for which this is publication LA-UR 08-0429. We thank S. Khochfar and M. Bertschik for their initial work on the implementation of the periodic boundaries. We thank Volker Springel for help with the Gadget-2 code as well as making a fix for a problem with the domain decomposition available to us. We thank Matthias Steinmetz for making the PPM results of the collapsing sphere test available to us. MW acknowledges support by Volkswagen Foundation under grant I/80 040. AFN wishes to thank UKAFF for financial support.

REFERENCES

Aarseth, S. J. 1963, MNRAS, 126, 223

—. 1999, PASP, 111, 1333

Athanassoula, E. 2003, MNRAS, 341, 1179

Athanassoula, E., Fady, E., Lambert, J. C., & Bosma, A. 2000, MNRAS, 314, 475

Attwood, R. E., Goodwin, S., P., & Whitworth, A. P. 2007, å, 464, 447

Balsara, D. S. 1990, PhD thesis, University of Illinois

—. 1995, J. Comp. Phys., 121, 357

Bate, M. R., Bonnell, I. A., & Price, N. M. 1995, MNRAS, 277, 362

Bate, M. R. & Burkert, A. 1997, MNRAS, 288, 1060

Bell, E. F., Naab, T., McIntosh, D. H., Somerville, R. S., Caldwell, J. A. R., Barden, M., Wolf, C., Rix, H.-W., Beckwith, S. V., Borch, A., Häussler, B., Heymans, C., Jahnke, K., Jogee, S., Koposov, S., Meisenheimer, K., Peng, C. Y., Sanchez, S. F., & Wisotzki, L. 2006, ApJ, 640, 241

Benz, W. 1988, Comp. Phys. Comm., 48, 97

Benz, W. 1990, in Numerical Modelling of Nonlinear Stellar Pulsations: Problems and Prospects, ed. J. R. Buchler (Kluwer, Dordrecht), 269

Benz, W. & Asphaug, E. 1995, Computer Physics Communications, 87, 253

Benz, W., Bowers, R. L., Cameron, A. G. W., & Press, W. H. 1990, ApJ, 348, 647

Berczik, P., Merritt, D., Spurzem, R., & Bischof, H.-P. 2006, ApJ, 642, L21

Bode, P. & Ostriker, J. P. 2003, ApJS, 145, 1

Bonnell, I. A., Clarke, C. J., & Bate, M. R. 2006, MNRAS, 368, 1296

Burkert, A. & Naab, T. 2005, MNRAS, 363, 597

Burkert, A., Naab, T., & Johansson, P. H. 2007, ArXiv e-prints, 710

Carraro, G., Lia, C., & Chiosi, C. 1998, MNRAS, 297, 1021

Dasyra, K. M., Tacconi, L. J., Davies, R. I., Genzel, R., Lutz, D., Naab, T., Burkert, A., Veilleux, S., & Sanders, D. B. 2006a, ApJ, 638, 745

Dasyra, K. M., Tacconi, L. J., Davies, R. I., Naab, T., Genzel, R., Lutz, D., Sturm, E., Baker, A. J., Veilleux, S., Sanders, D. B., & Burkert, A. 2006b, ApJ, 651, 835

Dave, R., Dubinski, J., & Hernquist, L. 1997, New Astronomy, 2, 277

Dehnen, W. 2001, MNRAS, 324, 273

Dilts, G. A. 1999, Int, J. for Num. Meth. in Eng., 44, 1115

—. 2000, Int, J. for Num. Meth. in Eng., 48, 1503

Dolag, K., Vazza, F., Brunetti, G., & Tormen, G. 2005, MNRAS, 364, 753

Efstathiou, G., Davis, M., White, S. D. M., & Frenk, C. S. 1985, ApJS, 57, 241

Evrard, A. E. 1988, MNRAS, 235, 911

Ewald, P. P. 1921, Ann. Phys., 64, 253

Ewell, M. W. 1988, PhD thesis, Princeton University

Fehlberg, E. 1968, NASA T.R., 287

Fletcher, C. A. J. 1997, Computational Techniques for Fluid Dynamics, second edition edn. (Springer-Verlag)

Frigo, M. & Johnson, S. G. 2005, Proceedings of the IEEE, 93, 216, special issue on "Program Generation, Optimization, and Platform Adaptation"

Fukushige, T., Ito, T., Makino, J., Ebisuzaki, T., Sugimoto, D., & Umemura, M. 1991, PASJ, 43, 841

Fukushige, T., Makino, J., & Kawai, A. 2005, PASJ, 57, 1009

Gingold, R. A. & Monaghan, J. J. 1977, MNRAS, 181, 375

—. 1982, J. Comp. Phys., 46, 429

Hernquist, L., Bouchet, F. R., & Suto, Y. 1991, ApJS, 75, 231

Hernquist, L. & Katz, N. 1989, ApJS, 70, 419

Hockney, R. W. & Eastwood, J. W. 1981, Computer Simulations Using Particles (McGraw-Hill Inc.)

Hockney, R. W. & Hohl, F. 1969, AJ, 74, 1102

Hultman, J. & Källander, D. 1997, A&A, 324, 534

Ito, T., Ebisuzaki, T., Makino, J., & Sugimoto, D. 1991, PASJ, 43, 547

Jackson, J. D. 1975, Classical Electrodynamics (Chichester, Toronto: John Wiley & Sons)

Jesseit, R., Naab, T., & Burkert, A. 2005, MNRAS, 360, 1185

Jesseit, R., Naab, T., Peletier, R. F., & Burkert, A. 2007, MNRAS, 376, 997

Kawai, A., Fukushige, T., Taiji, M., Makino, J., & Sugimoto, D. 2000, PASJ, 52, 659

Khochfar, S. & Burkert, A. 2006, A&A, 445, 403

Klessen, R. 1997, MNRAS, 292, 11

Kutta, W. 1901, Zeitschrift für Mathematik und Physik, 46, 435

Lattanzio, J. C., Monaghan, J. J., Pongracic, H., & Schwarz, M. P. 1986, SIAM J. Scient. Comp., 7, 591

Levison, H. F. & Duncan, M. J. 1994, Icarus, 108, 18

Lia, C. & Carraro, G. 2000, MNRAS, 314, 145

Lombardi, J. C., Sills, A., Rasio, F. A., & Shapiro, S. L. 1999, J. Comp. Phys., 152, 687

Lucy, L. B. 1977, AJ, 82, 1013

Makino, J., Fukushige, T., Koga, M., & Namura, K. 2003, PASJ, 55, 1163

Makino, J., Ito, T., & Ebisuzaki, T. 1990, PASJ, 42, 717

Makino, J. & Taiji, M. 1998, Scientific Simulations with Special-Purpose Computers : The GRAPE Systems (Chichester , Toronto: John Wiley & Sons)

Makino, J., Taiji, M., Ebisuzaki, T., & Sugimoto, D. 1997, ApJ, 480, 432

Merritt, D. 1996, AJ, 111, 2462

Merritt, D. & Szell, A. 2006, ApJ, 648, 890

Monaghan, J. J. 1985, Comp. Phys. Rep., 3, 71

—. 1988, Comp. Phys. Comm., 48, 89

—. 1989, J. Comp. Phys., 82, 1

—. 1992, ARA&A, 30, 543

Monaghan, J. J. & Gingold, R. A. 1983, J. Comp. Phys., 52, 374

Monaghan, J. J. & Lattanzio, J. C. 1985, A&A, 149, 135

Morris, J. M. & Monaghan, J. J. 1997, J. Comp. Phys., 136, 41

Naab, T. & Burkert, A. 2003, ApJ, 597, 893

Naab, T., Burkert, A., Johansson, P. H., & Jesseit, R. 2007, ArXiv e-prints, 709

Naab, T., Jesseit, R., & Burkert, A. 2006a, MNRAS, 372, 839

Naab, T., Khochfar, S., & Burkert, A. 2006b, ApJ, 636, L81

Naab, T. & Trujillo, I. 2006, MNRAS, 369, 625

Nelson, A. F. 2006, MNRAS, 373, 1039

Nelson, A. F., Benz, W., Adams, F. C., & Arnett, D. 1998, ApJ, 502, 342

Nelson, A. F., Benz, W., & Ruzmaikina, T. V. 2000, ApJ, 529, 357

Nelson, A. F., Wetzstein, M., & Naab, T. 2006, ApJ, 0, 0

Nelson, R. P. & Papaloizou, J. C. B. 1994, MNRAS, 270, 1

Okumura, S. K., Makino, J., Ebisuzaki, T., Fukushige, T., Ito, T., Sugimoto, D., Hashimoto, E., Tomida, K., & Miyakawa, N. 1993, PASJ, 45, 329

O'Shea, B. W., Bryan, G., Bordner, J., Norman, M. L., Abel, T., Harkness, R., & Kritsuk, A. 2004, in Adaptive Mesh Refinement - Theory and Applications, ed. T. Plewa, t. Linde, & V. Weirs, Lecture Notes in Computational Science and Engineering (Springer-Verlag, Berlin Heidelberg New York)

O'Shea, B. W., Nagamine, K., Springel, V., Hernquist, L., & Norman, M. L. 2005, ApJS, 160, 1

Portegies Zwart, S. F., Baumgardt, H., McMillan, S. L. W., Makino, J., Hut, P., & Ebisuzaki, T. 2006, ApJ, 641, 319

Porter, D. 1985, PhD thesis, University of California, Berkeley

Press, W., Teukolsky, S. A., Vetterling, W., & Flannery, B. P. 1992, Numerical Recipes, 2nd edn. (Cambridge University Press, Cambridge)

Price, D. J. & Monaghan, J. J. 2004, MNRAS, 348, 139

—. 2007, MNRAS, 374, 1347

Rasio, F. A. & Shapiro, S. L. 1991, ApJ, 377, 559

Romeo, A. B. 1997, A&A, 324, 523

Rosswog, S., Davies, M. B., Thielemann, F.-K., & Piran, T. 2000, A&A, 360, 171

Ryu, D., Ostriker, J. P., Kang, H., & Cen, R. 1993, ApJ, 414, 1

Salmon, J. K. & Warren, M. S. 1994, J. Comp. Phys., 111, 136

Springel, V. 2005, MNRAS, 364, 1105

Springel, V. & Hernquist, L. 2002, MNRAS, 333, 649

Springel, V., Yoshida, N., & White, S. D. M. 2001, New Astronomy, 6, 79

Steinmetz, M. 1996, MNRAS, 278, 1005

Steinmetz, M. & Müller, E. 1993, A&A, 268, 391

Stone, J. M. & Norman, M. L. 1992, ApJS, 80, 753

Sugimoto, D., Chikada, Y., Makino, J., Ito, T., Ebisuzaki, T., & Umemura, M. 1990, Nature, 345, 33

Thacker, R. J., Tittley, E. R., Pearce, F. R., Couchman, H. M. P., & Thomas, P. A. 2000, MNRAS, 319, 619

Thomas, J., Jesseit, R., Naab, T., Saglia, R. P., Burkert, A., & Bender, R. 2007, MNRAS, 381, 1672

Wadsley, J. W., Stadel, J., & Quinn, T. 2004, New Astronomy, 9, 137

Wetzstein, M., Naab, T., & Burkert, A. 2007, MNRAS, 375, 805

This 2-column preprint was prepared with the AAS LaTeX macros v5.2.