1 HIGH PERFORMANCE COMPUTATIONAL ASTROPHYSICS WITH PKDGRAV/GASOLINE

Joachim Stadel, James Wadsley and Derek C. Richardson

University of Washington, Seattle, USA

{stadel, wadsley, dcr}@astro.washington.edu

http://www-hpcc.astro.washington.edu/

Abstract:

We describe the design and implementation of the pkdgrav (gravity) and gasoline (gravity plus hydrodynamics) astrophysical N-body codes. We present a diverse spectrum of applications using these codes, from cosmological N-body simulations including gas physics, to solar system formation and granular dynamics simulations. Keywords: Parallel Algorithms, Gravity, N-Body, Hydrodynamics, Smoothed Particle Hydrodynamics, Cosmology, Planet Formation

1.1 INTRODUCTION

Historically the N-body problem in astrophysics dealt with the evolution of the Solar System under the influence of gravity. Over the last several decades most of the attention and computing resources in astrophysics has been focused on cosmological N-body simulations with larger and larger numbers of particles. These are indeed some of the largest simulations to be performed in all of computational science. However, modern questions about the stability and dynamical chaos of the Solar System require simulations using many billions of timesteps and sophisticated integration techniques, making this old problem a present day computational challenge as well. Between this big and small N there is a diverse range of astrophysical problems open to N-body simulations, particularly when gas and collisional physics are incorporated into simulation codes. We show this range in Figure 1.1, where T is the number of timesteps that a present day "state-of-the-art" simulation would take.

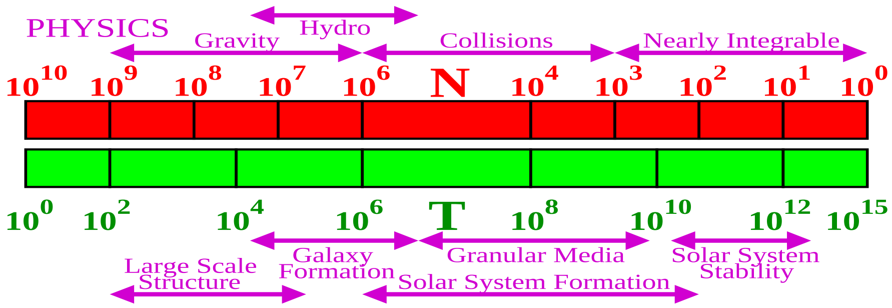

Figure 1.1 The spectrum of astrophysical N-body simulations, showing increasing number of particles, N, and decreasing number of timesteps, T. The product of N and T is roughly proportional to computational work, the combined scales corresponding to present day "state-of-the-art" simulations. Shown are the various ranges of physics that addressed by PKDGRAV/GASOLINE. Nearly Integrable: systems where some form of perturbation theory can be used in the calculation, such as protoplanets orbiting the Sun. Collisions: systems where collisions between bodies are most relevant to the creation or evolution of structure. 'Hydro: systems where gas physics (along with gravity) is used, mainly to address the problem of galaxy formation. Gravity: systems where gravity alone dictates structure. On the bottom of the figure the ranges for some of the scientific questions addressed by our code are shown.

Parallel computers have been very effective in this field despite the difficulties presented by the non-uniform, spatially and temporally adaptive data structures required in handling this wide range of applications. A very small set of generic N-body algorithms are sufficient, reducing the complexity and simplifying the addition of new physical effects. Our code, PKDGRAV (GASOLINE being the name of its gas physics extension), is used for all of these applications, all relying on the same generic data structure and algorithmic core.

This paper is organized as follows. The next section discusses design issues for PKDGRAV which are generally relevant to all applications, first looking at serial processing design issues and following with parallel computing considerations. The next three sections deal with specifics for three different application of PKDGRAV. In Section 3 we discuss Cosmological N-body simulations. In Section 4 we focus on simulations including gas physics via Smoothed Particle Hydrodynamics (SPH). Finally, Section 5 discusses the addition of collisional physics to the code and the new problems that can be addressed by this.

1.2 DESIGN STRATEGY

Central to the design of PKDGRAV was code portability. PKDGRAV is written in C allowing the code to be compiled for any architecture. No calls are made in the code to any particular parallel message passing or shared memory library, instead calls are made to small library that was designed to hide the architecture, providing higher level primitives needed by the code. This layer of abstraction has allowed PKDGRAV to run efficiently on both distributed and shared memory parallel computers and simplified the migration to new architectures. Some aspects of this MDL library are discussed in more detail below. Modular design was another principle goal. This has allowed new authors to add to and modify PKDGRAV with relative ease. Along with the use of CVS, parallel development of the code has been very successful and is evidenced by the fact that each of the authors of this paper (along with Thomas Quinn) have been significant developers of PKDGRAV.

Improvements in the performance and quality of N-Body simulations has been sought in four general areas: 1) faster calculation of forces; 2) multistepping, which is the reduction in number of time steps taken by particles in regions of the simulation where longer dynamical times are expected; 3) volume renormalization, where regions of interest are identified and populated with a greater density of particles than the surrounding volume (Katz *et al.* 94); 4) the use of parallel and vector supercomputing techniques. The need for rapid calculation of the gravitational accelerations has led to two basic approaches. The first uses grid techniques, relying mainly on the speed of FFT algorithms for the calculation of the gravitational field. This class includes the PM, P^3M (Hockney & Eastwood 88) and AP^3M (Couchman 92) algorithms. PKDGRAV falls into the second class known as tree codes (Barnes 86; Barnes & Hut 86; Greengard & Gropp 87; Greengard 88), which use multipole expansions within a hierarchical description of the mass distribution (a tree structure). Tree codes allow the mutual gravitational forces on N bodies to be calculated in $O(NlogN)$ time instead of the naive $O(N^2)$.

1.2.1 The Tree

Unlike the more traditional *oct-tree*, the central data structure used by PKDGRAV is a *spatial binary tree*. This binary tree forms a hierarchical representation of the mass distribution, of which the root node or cell encloses the entire simulation volume. The tree is built in a manner very similar to the quick-sort algorithm, always recursively bisecting the longest axis of each cell, starting with the rectangular root cell until 8 particles or fewer remain in a cell. Allowing a maximum of 8 particles at the leaf-cells (or *buckets*) reduces the storage required by the tree and provides near optimum amortization in the gravity calculation. Once the tree has been built we do a bottom-up pass starting from the buckets and proceeding to the root, calculating the center of mass and the multipole moments of each cell from the center of mass and moments of each of its two sub-cells. This build phase requires at most about 5% of the gravity calculation time.

1.2.2 Calculating Gravity

PKDGRAV calculates the gravitational accelerations using the well known *tree-walking* procedure of the Barnes-Hut algorithm (Barnes & Hut 86), except that

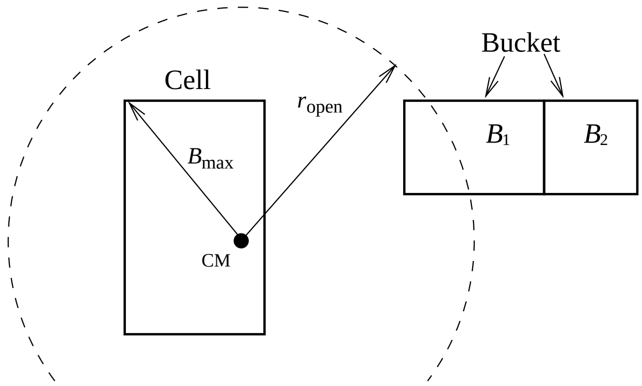

Figure 1.2 Opening radius for a cell in the tree, intersecting bucket B_1 and not bucket B_2. This cell is "opened" when walking the tree for B_1. When walking the tree for B_2, the cell will be added to the particle-cell interaction list of B_2.

it collects interactions for entire buckets rather than single particles. Thus, it amortizes the cost of tree traversal for a bucket, over all its particles.

In the tree building phase, PKDGRAV assigns to each cell of the tree an *opening radius* about its center-of-mass. This is defined as,

$$r_{\mathrm{open}} = \frac{2B_{\mathrm{max}}}{\sqrt{3}\,\theta} \tag{1.1}$$

where B_{max} is the maximum distance from a particle in the cell to the center-of-mass of the cell. θ is a user specified accuracy parameter which is similar to the traditional θ parameter of the Barnes-Hut code; notice that decreasing θ in Equation 1.1, increases r_{open}.

The opening radii are used in the *Walk* phase of the algorithm as follows: for each bucket B_i, PKDGRAV starts descending the tree, "opening" those cells whose r_{open} intersect with B_i (see Figure 1.2). If a cell is "opened," then PKDGRAV repeats the intersection-test with B_i for the cell's children. Otherwise, the cell is added to the *particle-cell interaction list* of B_i. When PKDGRAV reaches the leaves of the tree and a *bucket* B_j is opened, all of B_j's particles are added to the *particle-particle interaction* list of B_i. Once the tree has been traversed in this manner we can calculate the gravitational acceleration for each particle of B_i by evaluating the interactions specified in the two lists. PKDGRAV uses a 4th-order multipole expansion to calculate particle-cell interactions increasing the accuracy of the forces and improving floating point performance.

One disadvantage of tree codes is that they must deal with periodic boundary conditions explicitly, unlike grid codes where this aspect is taken care of implicitly. Although this adds complexity to any tree code, it is possible to incorporate periodic boundary conditions efficiently by using a 4th-order multipole approximation to the *Ewald* summation technique (Hernquist *et al.* 91; Ding *et al.* 92).

1.2.3 *Multistepping*

As the number of particles in an N-body simulation grows, so do the density contrasts. Hierarchical methods can follow extremely large dynamic ranges in densities at modest additional cost per force evaluation. However, large ranges in densities also imply a large range in time scales ($\propto 1/\sqrt{\text{density}}$). If we take the final state of a simulation and weight the computational work done on particles not uniformly but inversely with their natural timesteps, we find a potential gain of ~ 50. Temporal adaptivity is one of the last algorithmic areas where we can target an order of magnitude improvement. PKDGRAV uses an adaptive leapfrog integrator (Quinn, et.al. 1997), a method where particles are on adjustable individual timesteps.

The performance gains achievable by using a particular adaptive timestepping scheme are limited by the "fixed costs" required for any level of the timestep hierarchy. For example, a tree code requires that the tree is built for all the particles regardless of the number of particles requiring force evaluations. Indeed, for a standard particle mesh code, the forces at every grid point are either calculated or not. The only part of the force evaluation that is not part of the fixed cost would be the trivial interpolation of the force onto particles. In this case, essentially all the costs are fixed and no real gain is possible. In addition to these fixed costs, the method of determining the timestep level assigned to each particle can incur a significant cost and must also be accounted for. Finally, inefficiency in calculating forces for a few particles on modern pipelined, vector or parallel processors, can add an extra "hidden" cost to any adaptive timestepping scheme. The amortization gains of having several particles per bucket are lost if only one of them needs its force calculated. However, despite these limitations the multistepping speedups for gravity simulations are typically a factor of 4 or more. With future efforts focused on reducing or eliminating these fixed costs speedups closer to the theoretical upper limits should be realized. Since tree building is currently the dominant fixed cost limiting PKDGRAV, the idea of repairing only parts of the tree, instead of rebuilding it is being investigated.

1.2.4 *Neighbour Search*

Our calculations always treat of sets of particles with physical properties such as position, velocity, mass assigned to them. Neighbour searching is not only useful for analysis but is used during runtime for most applications of our code. Particularly it is used to perform *smoothing* operations: to get local smoothed estimates of physical quantities such as density and to find potential colliders in solar system scale simulations. Local smoothed estimates are the basis of Smoothed Particle Hydrodynamics (SPH) and density is a useful timestep estimator for Gravity only simulations.

Fundamentally this is a *k-Nearest Neighbour* problem. For the applications discussed below k is normally in the range 6-64. From physical considerations of momentum and energy conservation we prefer symmetrized interactions so that all particles are mutual neighbours: a given particle pair undergoes an exchange type interaction if either is a k-nearest neighbour of the other. For a single particle this can be thought of as a two part process, a *gather* operation from its k-Nearest Neighbours and a *scatter* operation from particles for which it is a k-nearest neighbour.

A general smoothed estimate for some quantity f at particles i given particles j at positions $\vec{r}_j$ takes the form:

$$f_i^{\rm smoothed} = \sum_{j=1}^{n} f_j A_{ij} W_{ij}(\vec{r}_i - \vec{r}_j, h_i, h_j), \tag{1.2}$$

where W is symmetric *kernel function* with compact support, h_j is a length indicative of the range of interaction of particle j and A_{ij} is symmetrized normalizing term (see section 1.4 for a specific example). A symmetric W may be achieved by assuming $W = 1/2W(|\vec{r}_i - \vec{r}_j|/h_i) + 1/2W(|\vec{r}_i - \vec{r}_j|/h_j)$. Other methods for symmetrizing W require similar neighbour finding solutions. The first and second contributions in W may be obtained independently via gather and scatter operations to particle i respectively.

When all the particles are considered at once a single round of gather operations accumulates the symmetrized interactions however a small *active* subset is more efficiently treated if only inactive particles with active k-nearest neighbours are tested: the *k-Inverse Nearest Neighbour* problem. Active in this context means requiring an update to dynamical quantities such as temperature or velocities based on interactions with other particles. For symmetrized interactions an active particle always gathers and scatters whereas an inactive one may scatter or contribute nothing. For practical simulations active subsets of particles are chosen by timestep. The number of particles in the active subsets range from a few particles out of millions to the nearly the entire set and they are usually spatially clumped.

Grids and linked lists have been used to locate neighbours however current applications with large spatial dynamic ranges benefit more from the use of tree structures. We employ a binary tree structure built by splitting the largest spatial dimension of the remaining particles bounding box recursively. The leaf nodes or buckets contain no more than $n_{\rm Bucket}$ particles. Setting $n_{\rm Bucket} = 8$ limits the tree size and is efficient when searching for of order 32 neighbours.

To find the k-nearest neighbour list for particle i we open the bucket that contains particle i and continue from it up the tree until the particles accumulated fill a *priority queue* of length k built using the distance $|\vec{r}_i - \vec{r}_j|$ from the particle i. The tree is walked to find and open all buckets that intersect a ball with radius equal to the largest distance in the queue and the contained particles distances compared to the first element in the queue (with the largest distance) and the closest retained. When the walk is complete the queued particles are the k-nearest neighbours. When one of the other particles in the queue needs to find its neighbours it can rebuild the queue to use as an efficient starting guess. The ball radius $f_{\rm Ball}$ of the kth-nearest neighbour for each particle is stored so the operation may be repeated efficiently (in half the cpu time) with a simple tree walk, ball gather as long as the particle distribution is unchanged.

When accumulating symmetrized interactions for active subsets of the particles we need to accumulate scattering interactions efficiently. Physical simulations are constrained to evolve at a stable pace and thus relative particle positions change slowly. To find the k-inverse nearest neighbours we solve the nearest neighbour problem once at the start and estimate an upper bound on the new ball radius for each particle $(1+\epsilon)f_{\rm Ball}$ with $\epsilon \sim 0.1$ estimated from particle motions. For each node in the tree we calculate the bounding box of the particles contained treated as spheres with radius $(1+\epsilon)f_{\rm Ball}$. If a given particle intersects this bounding box it may be an inverse nearest neighbour of one of the particles contained in that node. We can thus walk the tree to accumulate scatter interactions. For typical distributions encountered in cosmological applications a full gather operation on all particles is comparably efficient if more than 10-20% of the particles are active. Perfect inverse nearest neighbour finding without testing all particles has been investigated (Tjaden & Anderson 00) however in 3 dimensions in excess of 20 k candidate inverse neighbours must be tested.

Current development aims to reduce the cpu costs of neighbour finding by accumulating prospective neighbours for entire nodes at once in a similar manner to the gravity tree walking.

Promising techniques for out Neighbour finding applications include computational geometric methods such as the Delaunay tetrahedralization of the points and walking the tetrahedra for neighbours. Part of the appeal of this technique is the prospect of efficient repair the the Delaunay tetrahedra using increment methods that scale as the number of actively moving particles (Joe 89).

1.2.5 Parallel Design

Achieving effective parallelism requires that work be divided equally amongst the processors in a way which minimizes interprocessor communication during the gravity calculation. Since we only need a crude representation for distant mass, the concept of data locality translates directly into spatial locality within the simulation. Each particle can be assigned a *work-factor*, proportional to the cost of calculating its gravitational acceleration in the prior time-step. Therefore, during domain decomposition, we divide the particles into spatially local regions of approximately equal work.

Experience has shown that using a data structure for the domain decomposition that does not coincide with the hierarchical tree for gravity calculation, leads to poor memory scaling with number of processors and/or tedious bookkeeping. That is the case, for instance, when using an Orthogonal Recursive Bisection (ORB) tree for domain decomposition and an oct-tree for gravity.

Current domain decomposition techniques for the oct-tree case involve forming "costzones," that is, processor domains out of localized sets of oct-tree cells (Singh 93), or "hashed oct-trees" (Warren & Salmon 95). PKDGRAV uses the ORB tree structure to represent the domain decomposition of the simulation volume. The ORB structure is completely compatible with the binary tree structure used for the gravity calculation. A root finder is used to recursively subdivide the simulation volume so that the sums of the work-factors in each processor domain are equal. Once this has been done, each processor builds a local tree from the particles within its domain. This entire domain decomposition and tree building process are fully parallelizable and incur negligible cost to the overall gravity calculation.

A small library of high level functions called MDL (Machine Dependent Layer) handles all parallel aspects of the code. This keeps the main gravity code architecture-independent and simplifies porting. For example, MDL provides a memory swapping primitive to move particles between processors during domain decomposition. Furthermore, MDL provides memory sharing primitives allowing local arrays of data to be visible to all processors. These primitives support access to non-local cells and particles during the *Walk* phase. In particular, a procedure called `mdlAquire` can be used to request and receive non-local data by providing an index into a non-local array, and an identifier for the processor that owns that array. On distributed memory systems MDL uses a software cache of recently accessed data greatly reducing network traffic. All the message passing complexity to maintain this cache is hidden from the main code. Some architectures allow highly optimized implementations of this memory sharing primitive, particularly those supporting one-sided communications such as the Cray-T3E.

1.3 GRAVITY AND APPLICATIONS

The isotropy of the microwave background radiation tells us that the early universe was exceptionally smooth, with density fluctuations smaller than 10^{-2} to 10^{-3} times the average density of the universe. However, the density within galaxies and clusters of galaxies today is on the order of 10^5 and $10^2 - 10^3$ times the average density of the universe respectively. Moreover, the galaxies have arranged themselves into very large coherent structures, the conspicuous walls and voids seen in redshift surveys. This raises the question of how the universe got from its early smooth state, to the present day wealth of structure we observe. It is now clear that gravitational instability causes the small "seed" fluctuations in the early universe to collapse, forming ever larger density contrasts. The statistics of the resulting structures are very sensitive to the power spectrum of the initial fluctuations, to the average density in the universe, and to the nature of the, as yet unknown, dark matter which is the dominant component of matter in the universe. It is because of this that the study of large scale structure formation is a cornerstone of modern cosmology.

The approach taken is to evolve an initial power spectrum of fluctuations under the influence of gravitational interactions to a present day distribution of matter, and then comparing this to the present day distribution of galaxies and clusters of galaxies. In this way it is possible to constrain the initial power spectrum of fluctuations and the cosmological model under which they grow. However, this comparison is difficult, since observations do not provide complete information on the distribution of galaxies and the fact that galaxies are not perfect tracers of the underlying mass distribution.

Furthermore, the fluctuations on smaller scales quickly grow to the point where there is significant nonlinear evolution and nonlinear coupling between different scales. This limits the usefulness of a purely analytical description of structure formation and large numerical simulations have become the principle theoretical tool in this field. These simulations are among the most expensive in all of science because they must follow the evolution of a fully self gravitating system which forms density contrasts with a dynamic range of over 6 orders of magnitude. The use of parallel supercomputers has vastly expanded the scope of investigations and the level of realism achievable with numerical simulations in the study of structure formation.

1.4 HYDRODYNAMICS AND APPLICATIONS

1.4.1 Eulerian versus Lagrangian Methods in Cosmology

Current cosmological models favour the existence of a dark collisionless component that is most of the mass and thus controls structure formation via its gravity. Gas is dynamically dominant only in highly dissipative regimes such as the star forming regions of galaxies.

Basic Eulerian cosmological simulations use a grid with fixed comoving resolution for the gas phase and rely upon particles to efficiently model the complex phase space of the collisionless dark matter with a particle-mesh scheme for the gravity. The mesh is essential to include gridded gas self-gravity consistently. This means that the dark matter spatial resolution is never better than 2 grid cells (Hockney & Eastwood 88) and thus the gravitational potentials are determined with constant mass resolution and poor spatial resolution compared to tree methods. Adaptive refinement for dark matter is not possible because the smallest scales are most non-linear and thus new small scale waves can only be correctly introduced in the initial linear regime.

For hydrodynamics leading Eulerian methods require less computation per resolution element (eg. 2 Cells to resolve a shock with PPM (Woodward & Collela 84)) than Lagrangian particle methods such as SPH (however see fig1.4.2). Langrangian means following the fluid flow at fixed mass resolution and thus highly collapsed objects of interest such as galaxies are resolved explicitly. *Adaptive Mesh Refinement* (AMR) can overcome this shortcoming of fixed grid schemes but will be limited by the mass resolution in the dark matter component. AMR requires padding zones around each refined stage however the hierachical nature of current cosmological models leads to fractal type solutions and thus the filling factor of dense fine structure plus padding tends to approach unity making the computation very expensive. In restricted cases

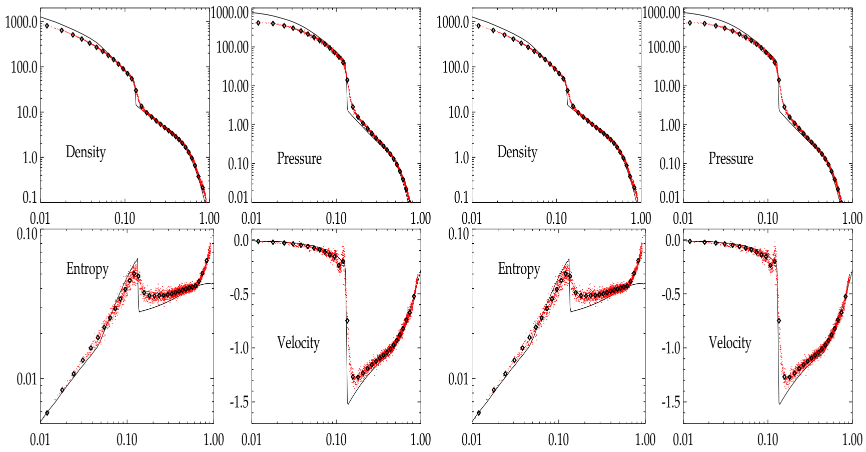

Figure 1.3 SPH Adiabatically collapsing sphere of gas approaching the self-similar solution. The diamonds indicate the one dimensional equivalent particle spacing and the solid lines are a very high resolution PPM solution. The left panel a good standard solution. The right panel has the SPH artificial velocity suppressed.

such as first stars and turbulence in galaxy cluster gas AMR has impressively resolved gas features(Bryan & Norman 97).

Thus a simple Lagrangian method such as SPH is still particularly useful in cosmology being well matched to the high resolution gravity methods and the nature of the problem.

1.4.2 Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics (SPH) is an approach to hydrodynamical modelling first developed by (Lucy 77; Gingold & Monaghan 77). It is a particle method that does not refer to grids for the calculation of hydrodynamical quantities: all forces and fluid properties are found on moving particles eliminating diffusive advective terms. The use of SPH for cosmological simulations required the development of variable smoothing to handle huge dynamic ranges (Hernquist & Katz 89; Couchman 91).

The basis of the SPH method is the representation and evolution of smoothly varying quantities whose value is only known at disordered discrete points in space. Estimates of density related physical quantities and gradients are generated using a kernel weighting function W. This characteristic led to SPH being described as a Monte Carlo type method (with $\mathcal{O}(1/\sqrt{N}$ errors) however it has been shown(Monaghan 85) that the method is more closely related to interpolation theory with errors $\mathcal{O}((lnN)^d/N)$, where d is the number of dimensions.

We employ a fairly standard implementation of the the hydrodynamic equations of motion for SPH(Monaghan 92). Density is calculated from a sum over particle masses m_j,

$$\rho_i = \sum_{j=1}^{n} m_j W_{ij}. \tag{1.3}$$

The momentum equation is expressed,

$$\frac{d\vec{v}_i}{dt} = -\sum_{j=1}^{n} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij} \right) \nabla_i W_{ij}, \tag{1.4}$$

where P_j is pressure, $\vec{v}_i$ velocity and the artificial viscosity term Π_{ij} is given by,

$$\Pi_{ij} = \begin{cases} \frac{-\alpha \frac{1}{2}(c_i+c_j)\mu_{ij}+\beta\mu_{ij}^2}{\frac{1}{2}(\rho_i+\rho j)} & \text{for } \vec{v}_{ij} \cdot \vec{r}_{ij} < 0, \\ 0 & \text{otherwise,} \end{cases} \tag{1.5}$$

$$\text{where } \mu_{ij} = \frac{h(\vec{v}_{ij} \cdot \vec{r}_{ij})}{\vec{r}_{ij}^{\,2} + 0.0025(h_i + h_j)^2}, \tag{1.6}$$

where $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$, $\vec{v}_{ij} = \vec{v}_i - \vec{v}_j$ and c_j is the sound speed. $\alpha = 1$ and $\beta = 2$ are coefficients we use for the terms representing bulk and Von Neumann-Richtmyer (high Mach number) viscosities respectively. We use a multiplicative switch (Balsara 95) $\frac{|\nabla\cdot\vec{v}|}{|\nabla\cdot\vec{v}|+|\nabla\times\vec{v}|}$ to suppress the viscosity in non-shocking, shearing environments.

To integrate the fluid equations we need to perform three smoothing operations where each subsequent smooth involves previous smoothed estimates. Firstly we accumulate density ρ, secondly the divergence $| \nabla \cdot \vec{v} |$ and curl $| \nabla \times \vec{v} |$ and finalley the pressure terms. Self-gravitating calculations also require a gravity calculation. Cpu costs for a typical calculation are spent mainly (90 %) on gravity however when treating small subsets due to multiple timesteps tree building and inverse neighbour finding become comparable.

In figure 1.4.2 we show results from a rigorous test of the code, the collapse under gravity of a spherical gas mass in three dimensions. The initial conditons for this calculation were a relaxed glass rather than a grid designed to be similar to a practical simulation. SPH manages to resolved the strong shock very well. The low level pre-shock entropy generation is a key concern for SPH in the cosmological context and an area of current work.

1.4.3 Galaxy Cluster Application

Cluster of galaxies are enormous systems ($\sim 10^{14}$ solar masses) that appear to be undergoing gravitational collapse at the present time. They contain huge masses of gas and their gravitational potential is sufficiently deep that infalling gas is shock heated to temperatures in excess of 10 million degrees. At these temperatures copius X-rays are emitted which we can detect with satellites. The process of gas infall and heating is complicated by the huge

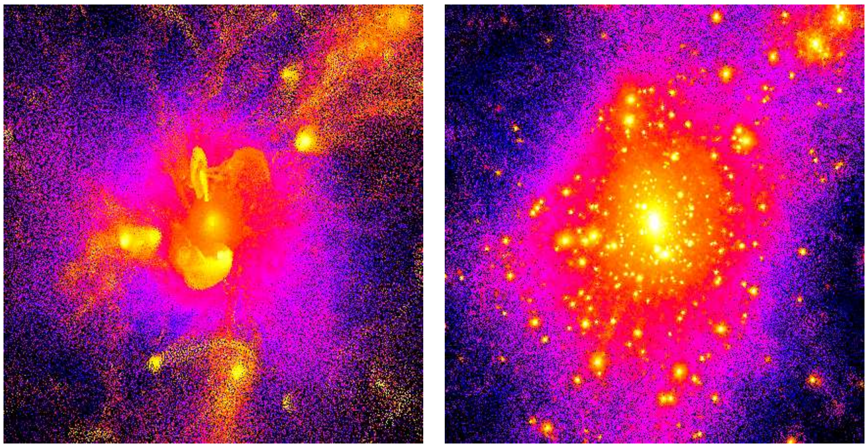

Figure 1.4 The gas entropy (light colours indicating lower entropy) and the dark matter density distribution in a inner 26 million light year box centred on a 4×10^{14} solar mass galaxy cluster. Note the turbulent gas structure and low entropy of lumps.

amount of substructure in these systems including 1000's of galaxies. Clusters have been the focus of numerical simulations for many years. Being rare objects their abundance throughout cosmlogical time is a strong function of the overall cosmology. On a more detailed level we seek to understand and interpret the X-ray signals and determine whether the temperature, distribution and mass of gas is consistent with currently favoured models of the universe.

In figure 1.4 we show the interior 26 million light years of 2.3 million particle cluster simulated to the current epoch. The final state cluster contains 4×10^{14} solar masses of gravitating material within the virial (gravitationally relaxed) radius of 6.5 million light years. The simulation took 2000 major steps with as many as 256 substeps per major step. An average major step took 2000 seconds on a cluster of 12 450 MHz pentium III using LAM MPI. A single substep involving all the particles took 500 seconds. Thus without multiple timesteps and $\mathcal{O}(N_{\text{active}})$ scaling we would have required up to 2 orders of magnitude more computation.

The cluster is both dynamic and highly turbulent. Gas at the core of the cluster is densest and dominates the X-ray emission. This gas tends to have a low entropy value which allows it to remain dense and thus occupy the cluster centre. Increasing gas entropy leads to higher gas temperatures or lower densities. Entropy is produced in shocks in the late forming cluster and earlier in collapsing substructure components. We performed our simulations seeking to track that entropy production. With this resolution it is clear that entropy is produced early on in filamentary structures and this gas is later loosely distributed throughout the cluster. Over time the core is disrupted by infalling lumps which shock the core gas directly however infalling lumps also transport entropy to the core. The net effect is a gradual rise in the entropy of the core which remains at a much lower level than the surrounding cluster gas. Remaining questions include the role of galactic outflows in contributing high entropy gas to the cluster.

1.5 COLLISIONS AND APPLICATIONS

There are many interesting dynamics problems in which surface contact between macroscopic particles is of comparable or greater importance than interparticle gravity. Familiar examples range from the formation of planets by "planetesimal" accretion to the dynamics of sandpiles and other granular media. Modelling collisions however poses unique challenges because the timescales involved can be tiny compared to the dynamical time of the system.

For many N-body applications (e.g. cosmology, galactic dynamics, etc.) single particles represent entire mass systems so their mutual interactions are "softened" to reduce the unphysical effect of two-body relaxation. But for most Solar System problems the bodies are fully resolved and therefore scattering and physical collisions are important. In `pkdgrav` collisions are predicted by searching for particle neighbours and extrapolating trajectories to see whether any two particles come into contact over a given time interval. In the second-order leapfrog scheme, it is natural to perform the search at the beginning of the linear "drift" step, when the particle velocities are kept fixed. The time to collision is then a simple quadratic:

$$\delta t = -\frac{(\mathbf{r}\cdot\mathbf{v})}{v^2}\left\{1 \pm \sqrt{1 - \left[\frac{r^2 - (R_1 + R_2)^2}{(\mathbf{r}\cdot\mathbf{v})^2}\right] v^2}\right\}, \tag{1.7}$$

where $\mathbf{r}$ and $\mathbf{v}$ are the relative position and velocity and R_1 and R_2 are the particle radii (we restrict ourselves to homogenous solid spherical particles). The sign ambiguity is resolved by choosing the smallest positive value of δt.

The fast neighbour-search scheme outlined above is used to generate the list of particles to check. In our special case the search boundaries can take into account the maximum possible drift of the particles to ensure no collisions are missed.

1.5.1 Planet Formation

The "planetesimal hypothesis" of planet formation (e.g. (Lissauer 93)) states that planets form via the pairwise accretion of smaller bodies, the planetesimals. The current thinking is that once km-sized bodies have formed, a runaway growth stage is entered in which a few protoplanets detach from the mass distribution. Once the protoplanets deplete material from their immediate environments they enter the stage of slow long-range mutual perturbations, eventually colliding with one another until there are only a few planets left. There are many poorly understood details in this general scheme, and numerical simulations are increasingly seen as the best way to understand them.

There are many possible outcomes following the collision of two planetesimals, e.g. merging, bouncing, cratering, or fragmentation. The outcome depends primarily on the relative impact energy $\frac{1}{2}\mu v^2$, where μ is the reduced mass. Often however it is assumed the particles simply merge on contact to form a new spherical body of the same density if the encounter speed is less than the mutual escape speed (and the spin of the combined mass is not unrealistically high owing to a grazing encounter). Otherwise the planetesimals bounce off with some prescribed dissipation (see (Richardson 94) for the general collision equations, including spin terms).

This was the procedure adopted for the $N = 10^6$ planetesimal run illustrated in Fig. 1.5. In this model the planetesimals were placed in orbits ranging from just outside the present-day location of Venus to a point near the middle of the present-day asteroid belt, with a surface density distribution that decreases with distance. An already-formed Jupiter was added to perturb the disk and determine whether growth is suppressed in the asteroid belt (it is too early to tell yet with this model). The run took approximately 200 wallclock hours to complete 1000 yr of integration (in fixed 0.01-yr timesteps) using a 300-MHz Cray T3E with 128 dedicated processors. This calculation represents the largest direct simulation of planetesimal evolution performed to date ((Richardson *et al.* 00)).

The computational challenge of simulating planet formation, at least in the inner Solar System, is not so much dealing with the collisions themselves (since most encounters lead to immediate mergers under the present idealized model) but the fact that it may take millions of dynamical times for a population of km-sized planetesimals to assemble into a handful of terrestrial planets. To be assured of adequately sampling all close encounters or collisions the timesteps need to be fractions of the dynamical time in some cases. This is where a stable adaptive multistepping method becomes crucial to make such simulations viable. We are currently experimenting with various algorithms that exploit the fact that most planetesimals make many orbits on only mildly perturbed elliptical paths before encountering a neighbour.

1.5.2 Rubble Piles

The simplifying assumptions regarding collision outcome during planet formation were made because so little is known about what really happens when two asteroid-sized bodies collide. Moreover, there is increasing evidence that km- to 100-km-sized bodies in the Solar System today are actually flying piles of rubble held together only by gravity (the evidence includes the spectacular breakup of Comet D/Shoemaker-Levy 9 at Jupiter, the remarkably low density of Asteroid 253 Mathilde, and the presence of doublet craters and crater chains in the inner Solar System; Cf. (Asphaug & Benz 94; Yeomans *et al.* 97; Richardson *al.* 98)). Simulations of colliding rubble piles may give insight into the formation and evolution of small bodies in the Solar System.

Figure 1.6 shows snapshots of a collision between two km-sized rubble piles simulated using `pkdgrav`. This example leads to a spherical remnant that is larger than either progenitor, yielding net growth. As the impact speed or impact angle is increased, however, the mass fraction retained is reduced,

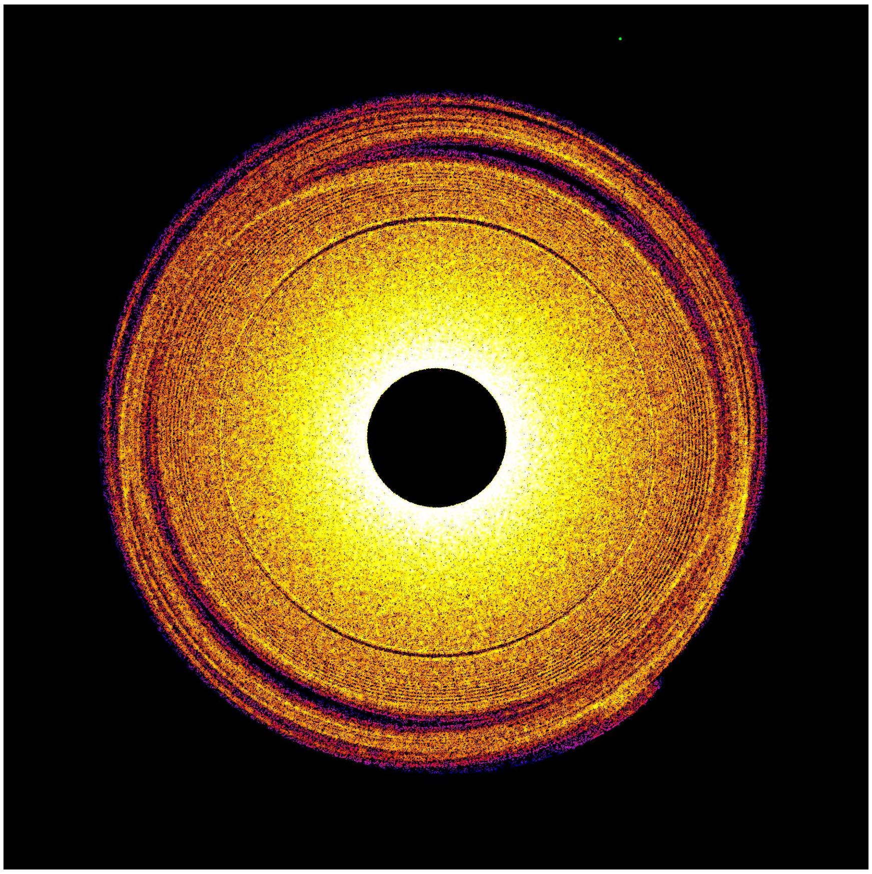

Figure 1.5 Mass density of a 10^6-planetesimal simulation after 250 yr. Bright shades represent regions of high density. The dot near the top is Jupiter. The gaps and spiral structures in the disk are associated with Jupiter mean-motion resonances.

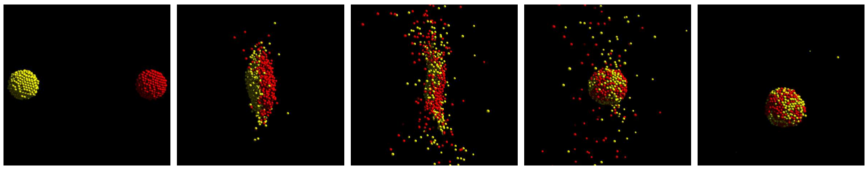

Figure 1.6 Snapshots of a low-speed collision between two km-sized rubble piles.

eventually leading to net erosion. In fact, for these simple models we find that only 35% of the time would two equal-size rubble piles on random impact trajectories grow in size ((Leinhardt *et al.* 00)). This apparent fragility may hamper planet growth in the early stage. We also find a variety of interesting remnant shapes, including pinwheels and dumbbells, reminiscent of some of the more exotic asteroids we see today, and some remnants with sizeable orbiting companions (a few percent of the total mass).

Rubble piles pose a different kind of computational challenge from planet formation since no mergers are allowed. This means that particles may be in close contact for a long time, resulting in millions of collision computations in typical runs of only a few thousand particles. Since the number of collisions per unit time increases geometrically with the number of particles, it rapidly becomes infeasible to model large ($N > 10^5$) rubble piles. Hence we generally restrict ourselves to $N < 10^4$. However, given the large parameter space of impact speed, impact angle, spin, etc. to explore, this problem is ideally suited to running on a Beowulf platform using a High-Throughput Computing environment such as that provided by `condor`.

1.5.3 Planetary Rings

Dense planetary rings can harbour special kinds of rubble-pile structures whimsically termed "dynamic ephemeral bodies," or DEBs ((Davis *et al.* 84)). Generally speaking planetary rings are located inside the Roche zone of the host planet, that is, the point at which a rubble pile would be torn apart by tidal forces. But if the density of particles is high enough, as it is in the bright A and B rings of Saturn, transient aggregates and "gravitational wakes" may form ((Salo 92; Richardson 94)). These structures can give rise to measurable brightness asymmetries in the rings, depending on the viewing angle, and may be directly observable when the Cassini spacecraft encounters Saturn in 2004.

Figure 1.7 shows a patch of Saturn's A ring modelled using `pkdgrav`. The calculation is carried out in a rotating frame using periodic (shearing!) boundary conditions since a direct model of the entire ring is still beyond reach. This is nevertheless by far the largest direct ring simulation performed to date, with $N \simeq 220,000$ particles(Porco *et al.* 99). Large N is necessary in order to adequately sample the steep particle size distribution while including at least 10 instability wavelengths in the azimuthal direction (typical particle sizes are 10 cm–5 m while the instability wavelengths are on the order of 50–100 m).

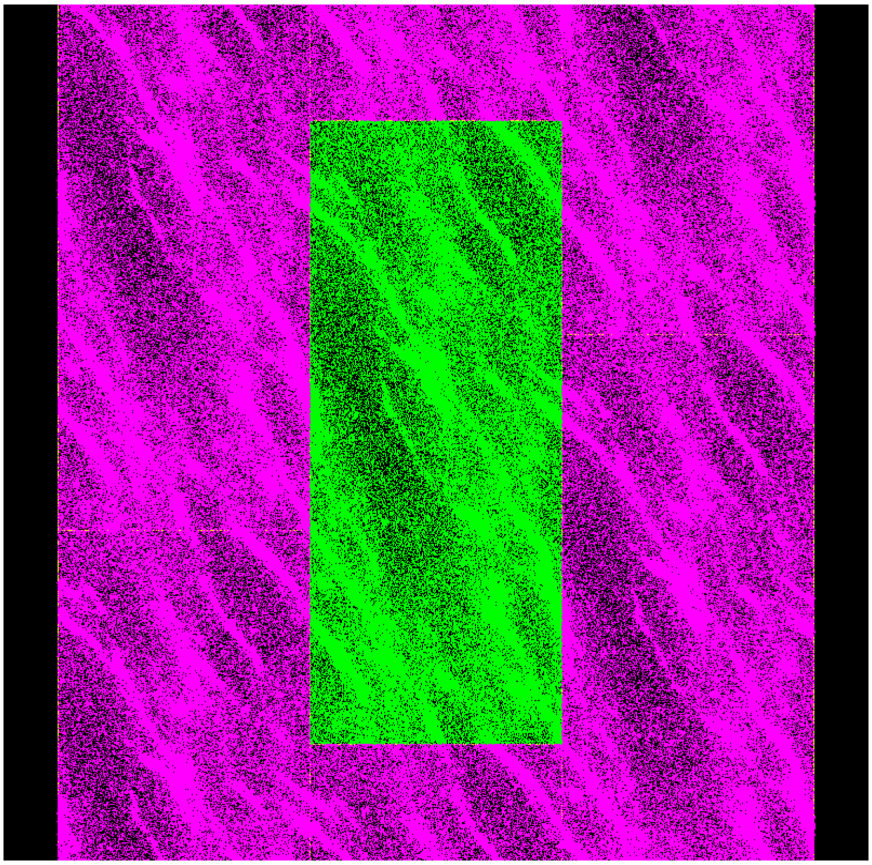

Figure 1.7 A patch of Saturn's A ring (light particles) after 10 orbits. Boundary conditions are provided by the surrounding replica "ghost patches" (dark particles). Wake formation on a scale of about 100 m is easily seen in this simulation, which was performed on the ARSC Cray T3E.

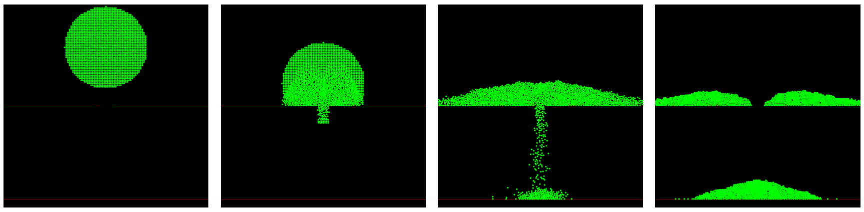

Figure 1.8 Snapshots of a 2D system of about 2,000 particles falling in a uniform gravity field onto a platform with a small opening.

Although the individual rubble pile structures that form can slow down the code considerably for the reasons alluded to earlier, their ephemeral nature makes this a relatively easier calculation. However, supercomputing resources are still required due to the large N.

1.5.4 Sandpiles

At the smallest size scales where gravitational and mechanical forces still dominate over electrostatic forces, we have so-called "sandpiles." The physics of such granular media are poorly understood, which presents an ideal opportunity for direct simulations. Moreover, the simulations can be compared against laboratory experiments, unlike the examples discussed earlier. This provides a means of testing the numerical code to see if, for example, we can reproduce the typical $\sim$ 25–35° slope angles of granular piles. We can also investigate mass segregation and packing efficiency (Cf. (Shinbrot & Muzzio 00) for a recent review of outstanding issues in granular dynamics).

These issues require special considerations, such as implementations of sliding friction, models of walls and other obstacles, and a means of avoiding "inelastic collapse." Inelastic collapse is a mathematical artifact stemming from the simplistic way dissipation is handled in most granular dynamics simulations (i.e. using a scalar coefficient of restitution) and that generally resting contact forces are not modelled. It can be shown that in systems involving a large number of repetitive collisions it is possible for the collision rate to spike to infinity (e.g. (McNamara & Young 94)). Under these circumstances the code either hangs or roundoff error causes particles to interpenetrate. One way of avoiding collapse is to turn off dissipation once the relative collision speed drops below a certain threshold set by the user. Hence the pile effectively has a minimum "temperature." This technique is used for self-gravitating rubble piles as well.

Figure 1.8 shows snapshots from a 2D simulation of a collection of particles falling onto a flat surface with a small opening;. Note the propagation of the granular shock wave following the initial contact. The final slopes are of order 20° for this monodispersive model with surface friction and perfectly round spheres. We are also experimenting with particle mixing in rotating cylinders which we plan to compare with laboratory experiments.

Acknowledgments

This work was supported in part by the Intel Technology 2000 program and a grant from the National Science Foundation.

References

Barnes, J. (1986). An efficient N-body algorithm for a fine-grain parallel computer. *The Use of Supercomputers in Stellar Dynamics*, pages 175–180. P. Hut and S. McMillan, eds, Springer Verlag.

Barnes, J., and Hut, P. (1986). A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324:446–449.

Couchman, H. (1992). Mesh-refined P^3M: a fast adaptive N-body algorithm. *Astrophys. J.*, **389**, 453–463.

Ding, H., Karasawa, N., and Goddard III, W. (1992). The reduced cell multipole method for coulomb interactions in periodic systems with million-atom unit cells. *Chemical Physics Letters*, **196**(1,2), 6–10.

Greengard, L. (1998). The rapid evaluation of potential fields in particle systems. PhD thesis, Yale University, Cambridge, Mass.

Greengard, L., and Gropp, W. (1987). A Parallel Version of the Fast Multipole Method. In Garry Rodrigue, editor, *Parallel Processing for Scientific Computing*, SIAM, 213–222.

Hernquist, L., Bouchet, F., and Suto, Y. (1991). Application of the Ewald Method to Cosmological N-Body Simulations. *The Astrophysical Journal Supplement Series*, **75**, 231–240.

Katz, N., Quinn, T., Bertschinger, E., and Gelb, J. (1994). Formation of quasars at high redshift. *Mon. Not. R. Astron. Soc.*, **270**, L71–L74.

Quinn, T., Katz, N., Stadel, J., and Lake, G. (1997). Time stepping n-body simulations. astro-ph/9710043.

Singh, J. (1993). *Parallel Hierarchical N-Body Methods and their Implications for Multiprocessors.* PhD thesis, Computer Systems Laboratory, Departments of Electrical Engineering and Computer Science, Stanford University.

Warren, M., and Salmon, J. (1995). A Parallel, Portable and Versatile Treecode. *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, SIAM, 319–324.

Balsara, D. S., "von Neumann stability analysis of smooth particle hydrodynamics–suggestions for optimal algorithms", *J. Computational Physics*, 121, 2, 357-372, 1995.

Bryan, G. & Norman, M. 1997, in *Computational Astrophysics*, Proc. 12th Kingston Conference, Halifax, Oct. 1996, ed. D. Clarke & M. West (PASP), p. 363

Couchman, H.M.P. 1991, *Astrophysical Journal Lett.*, 368, 23

Gingold, R.A. & Monaghan, J.J. 1977, *Monthly Notices R.A.S.*, 181, 375

Hernquist, L. & Katz, N. 1989, *Astrophysical Journal Supp.*, 70, 419

Hockney, R.W. & Eastwood, J.W. 1988, *Computer Simulation using Particles*, IOP Publishing

Joe, B. (1989), "Three-dimensional triangulations from local transformations", *SIAM J. Sci. Stat. Comput.*, 10, pp. 718-741.

Lucy, L. 1977, *Astronomical Journal*, 82, 1013

Monaghan, J.J. 1985, *Computer Physics Reports*, 3, 71

Monaghan, J.J. 1992, *Annual Reviews in Astronomy and Astrophysics*, 30, 543

Steinmentz, M. & Müller, E. 1993, *Astronomy and Astropysics*, 268, 391

Tjaden, B. C. & Anderson, R. "Inverse Nearest Neighbour Searching", *in preparation*

Woodward, P. & Collela, P. 1984, *Journal of Computational Physics*, 54, 115

Asphaug, E., and Benz, W. (1994). Density of Comet Shoemaker-Levy-9 deduced by modelling breakup of the parent rubble pile. *Nature* **370**, 120–124.

Davis, D. R., Weidenschilling, S. J., Chapman, C. R., and Greenberg, R. (1984). Saturn ring particles as dynamic ephemeral bodies. *Science* **224**, 744–747.

Leinhardt, Z. M., Richardson, D. C., and Quinn, T. (2000). Direct N-body simulations of rubble pile collisions. *Icarus*, in press.

Lissauer, J. J. (1993). Planet formation. *Annu. Rev. Astron. Astrophys.* **31**, 129–174.

McNamara, S., and Young, W. R. (1994). Inelastic collapse in two dimensions. *Phys. Rev. E* **50**, R28–R31.

Porco, C. C., Pantazopoulou, M. J., Richardson, D., Quinn, T., and Kehoe, T. J. J. (1999). Light scattering in planetary rings: The nature of Saturn's particle disk. *Bull. Am. Astr. Soc.* **31**, 1140.

Richardson, D. C. (1994). Tree code simulations of planetary rings. *Mon. Not. R. Astr. Soc.* **269**, 493–511.

Richardson, D. C., Bottke, W. F., Jr., and Love, S. G. (1998). Tidal distortion and disruption of Earth-crossing asteroids. *Icarus* **134**, 47–76.

Richardson, D. C., Quinn, T., Stadel, J., and Lake, G. (2000). Direct large-scale N-body simulations of planetesimal dynamics. *Icarus* **143**, 45–59.

Salo, H. (1992). Gravitational wakes in Saturn's rings. *Nature* **359**, 619–621.

Shinbrot, T., and Muzzio, F. J. (2000). Nonequilibrium patterns in granular mixing and segregation. *Physics Today* **53**, 25–30.

Yeomans, D. K., and 12 colleagues (1997). Estimating the mass of Asteroid 253 Mathilde from tracking data during the NEAR flyby. *Science* **279**, 2106–2109.