A VECTORIZED ALGORITHM FOR MOLECULAR DYNAMICS OF SHORT RANGE INTERACTING PARTICLES

VOLKHARD BUCHHOLTZ

and

THORSTEN PÖSCHEL

Humboldt–Universität zu Berlin, Institut für Theoretische Physik,

Unter den Linden 6, D–10099 Berlin, Germany

We report on a lattice based algorithm, completely vectorized for molecular dynamics simulations. Its algorithmic complexity is of the order $O(N)$, where N is the number of particles. The algorithm works very effectively when the particles have short range interaction, but it is applicable to each kind of interaction. The code was tested on a CRAY YMP EL in a simulation of flowing granular material.

Keywords: Algorithms; Vectorization; Granular Flow.

1. Introduction

The properties of systems consisting of many interacting particles have been subjects of interest to scientists for hundreds of years. Since the equations of motion do not allow for an analytical solution[1] one has to calculate the trajectories of the particles by integrating NEWTONs equation of motion numerically for each particle i:

$$\ddot{\vec{r}}_i = \frac{1}{m_i} \cdot \vec{F}_i(\vec{r}_j) \qquad (j = 1, \ldots, N) \tag{1}$$

This method is called molecular dynamics simulation. Molecular dynamics simulations have led to many important results, especially in fluid dynamics, solid state physics, polymer physics, and plasma physics.

There are many methods to integrate sets of ordinary differential equations.[2] One of the most common is the GEAR–predictor–corrector method[3] which we used in our simulations (see Appendix).

The main problem in molecular dynamics is the calculation of the forces acting upon each of the particles. The term $\vec{F}_i(\vec{r}_j)$ may be very difficult to determine, but even if it is not too difficult and the forces depend only on the pairwise distances of the particles, the algorithmic complexity is at least of the order $O(N^2)$ since each

particle may interact with each other. In fact the probability that two particles which are a large distance apart at time t, and interact within a small time interval $(t, t + \Delta t)$ (Δt is much larger than the integration step δt) is very low, due to the probability distribution of the particle's velocities. That means that even if there is a long range force as the electrostatic force there is a critical threshold for the distance beyond which the particles do not influence each other. Therefore one can neglect the interaction of those particles during some time interval Δt and one can use lists of neighbors j for each particle i containing the particles j which are closer to particle i than a given threshold.[4] The neighborhood–list has to be updated each time interval Δt. The asymptotic time–complexity, however, is not reduced because the calculation of the neighborhood list is still of the order $O(N^2)$. Nevertheless the calculation time reduces drastically. Unfortunately in the general case there is no simple way to vectorize a molecular dynamics code using neighborhood lists, e.g. Ref. 5. There are hierarchical force calculation algorithms of complexity $O(N \log(N))$ (Ref. 6), and even of complexity $O(N)$ (Ref. 7). These algorithms make use of fast multipole expansions, which cannot be applied to each type of particle interaction, as for the case of short range interaction, in particular if the force is not a steady function of the distance between two interacting particles.

The aim of this paper is to describe a completely vectorizable algorithm for molecular dynamics, which acts very effectively, provided the following preconditions hold:

1. There is no long range interaction between the particles.

2. The particles are allowed to deform only slightly.

3. The dispersion of the particle size is not too large.

4. The particle density, i.e. the number of particles per space unit is high.

These preconditions are provided in the molecular dynamics simulations of granular material, as we will demonstrate below, which are of special interest.

2. Description of the Algorithm

Initially, we assume a region, where particles are allowed to move. One can also think about a periodically continued region to simulate periodic boundary conditions. Now we define a square lattice which covers this region and its cell size is determined by the requirement that not more than one particle can reside per lattice cell. Obviously the cell size d is determined by the smallest particles and by the smallest distance d_{min} they can have during the simulation $d = d_{min}/\sqrt{2}$ (fig. 1).

Given this lattice our algorithm acts as follows:

1. Set the variables to their initial values.

2. Predict the positions and the time derivatives due to the predictor–corrector algorithm as described in the Appendix.
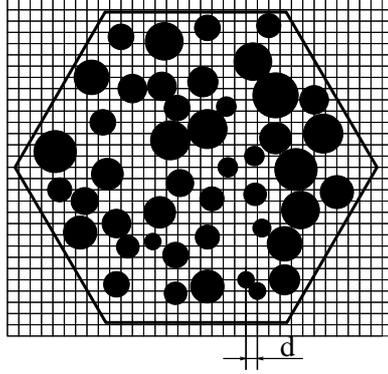
Figure 1: *The size of the lattice cell is determined by the smallest particles and their minimum distance during the simulation since there must not be more than one particle residing within each lattice cell at time t.*

3. Set up the lattice vectors:

$$\overline{Index}[i] = \left\{ \begin{array}{ll} \text{particle number} & \text{if there resides a particle} \\ 0 & \text{otherwise} \end{array} \right. \tag{2}$$

$$i \in [1, LMAX]$$

$$\overline{Test}[i] = \left\{ \begin{array}{ll} 1 & \text{if there is a particle} \\ 0 & \text{otherwise} \end{array} \right. \tag{3}$$

$$i \in [1, LMAX],$$

where $LMAX$ is the number of lattice cells.

The vector $\overline{Index}$ maps all observables which belong to the particles to corresponding lattice vectors, so that we get a set of vectors $\overline{x}$, $\overline{y}$, $\overline{F_x}$, $\overline{F_y}$, etc.

4. Calculate the forces acting upon the particles.
Because the particles are assumed to have short range interactions we only regard the interactions of a particle in cell (i, j) with particles which reside within cells (k, l), $(k \in [i-2, i+2], l \in [j-2, j+2], (k, l) \neq (i, j))$. Therefore we may define a mask (fig. 2) which describes the range of particle interactions. To determine the interaction of each particle with the others we move the centre of the mask through all sites of the lattice and calculate the interactions of the particles which might lie within the mask. The index i, $i \in [1, 24]$, points to the mask positions which interacts with the regarded cell. Hence the distances as well as all interesting values of the particles may be expressed by 24 vectors indexed by the numbers of the lattice cells.

$$\overline{dist}_i = \sqrt{\overline{\Delta x_i}\overline{\Delta x_i} + \overline{\Delta y_i}\overline{\Delta y_i}} \qquad i \in [1, 24] \tag{4}$$

with

$$\overline{\Delta x_i} = \overline{x} - \overline{x\{i\}} \tag{5}$$

$$\overline{\Delta y_i} = \overline{y} - \overline{y\{i\}}, \tag{6}$$

where the vector $\overline{x\{i\}}$ equals the vector $\overline{x}$ shifted due to the position of i according to the mask enumeration as given in fig. 2.

15	14	13	12	11
16	4	3	2	10
17	5	0	1	9
18	6	7	8	24
19	20	21	22	23

Figure 2: *Mask describing the area in the neighborhood of a particle, which resides in lattice cell 0.*

Now we define a set of vector variables $\overline{contact}_i$ $i \in [1, 24]$ which checks whether or not the particles touch each other.

$$\overline{contact}_i = \begin{cases} 1 & \text{if the particles touch each other} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$
$$i \in [1, 24].$$

With these variables the normal and shear forces read as:

$$\overline{F_n}_i = \overline{Test} \cdot \overline{Test\{i\}} \cdot \overline{contact}_i \cdot \mathcal{F}_n(0, i) \tag{8}$$

$$\overline{F_s}_i = \overline{Test} \cdot \overline{Test\{i\}} \cdot \overline{contact}_i \cdot \mathcal{F}_s(0, i), \tag{9}$$

where $\mathcal{F}_n(0, i)$ and $\mathcal{F}_s(0, i)$ denote the normal and shear forces between particles which possibly reside in mask position 0 and i ($i \in [1, 24]$).

The total forces and the momentum acting upon each particle are given by:

$$\overline{F_x} = \sum_{i=1}^{24} \frac{1}{\overline{dist}_i} \cdot (\overline{\Delta x_i} \cdot \overline{F_n}_i + \overline{\Delta y_i} \cdot \overline{F_s}_i) \tag{10}$$

$$\overline{F_y} = \sum_{i=1}^{24} \frac{1}{\overline{dist}_i} \cdot (\overline{\Delta y_i} \cdot \overline{F_n}_i - \overline{\Delta x_i} \cdot \overline{F_s}_i) \tag{11}$$

$$\overline{M} = -\sum_{i=1}^{24} \overline{r} \cdot \overline{F_s}_i \qquad . \tag{12}$$

5. Correct the predicted values due to the GEAR–algorithm as described in the
   Appendix.

6. Increase the time $t := t + \delta t$.
   Proceed with step 2.

Assuming that the particles do not penetrate each other more than 10% of their
radii, this algorithm works correctly if the ratio of the smallest and the largest radii
does not exceed $1 : \frac{1.8}{\sqrt{2}}$. The cell size d has to be set to the value of the maximum
radius ($d = r_{max}$).
At each time–step one can extract all physical observables of interest.

3. Efficiency of the Algorithm

To evaluate the efficiency of the algorithm it is necessary to determine the number of operations NOP of each step described in Section 2 and the corresponding
vector length n. Thereby we disregard the possibility of chaining. Given a pipeline
depth D, one elementary vector operation requires a time of $n + D$ clock periods
instead of $n \cdot D$ which is the required time to process the equivalent code on a scalar
machine.

For each part of the algorithm we get the results summarized in the following
table.

Part of algorithm	NOP	n	estimated time
Predictor	59	NP	$59 \cdot (NP + D)$
Index	12	NP	$12 \cdot (NP + D)$
Forces	1656	$LMAX$	$1656 \cdot (LMAX + D)$
Corrector	54	NP	$54 \cdot (NP + D)$

NP number of particles
$LMAX$ number of lattice cells

One iteration step of the algorithm needs the time:
$t = 125 \cdot (NP + D) + 1656 \cdot (LMAX + D)$ clock periods.
That means the time depends strongly on the number of lattice cells but hardly
on the number of particles. Fig. 3 shows that the time of the vectorized algorithm
does not vary as rapidly with increasing number of particles as the time of the
scalar algorithm. Figure 4 shows the calculation time (normalized by the number
of particles) as a function of the number of particles for different particle densities
$\rho = \frac{NP}{LMAX}$. The required time for the neighborhood–list method does not depend
on the particle density, because it works without the lattice.

The density is obviously the limiting factor for the efficiency of the algorithm.
To be more effective than the neighborhood–list method, the number of particles
must increase with the reciprocal of the parameter ρ. Therefore the algorithm is
constrained to problems with high particle density or to very large systems with a
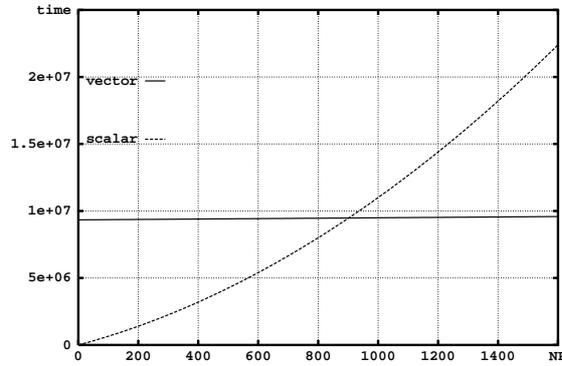lower density.

Figure 3: *The processing time for vectorized calculations depends very weakly on the number of particles while the time for the scalar neighborhood–list method rises as N^2. The pipeline depth D was assumed to be 20.*
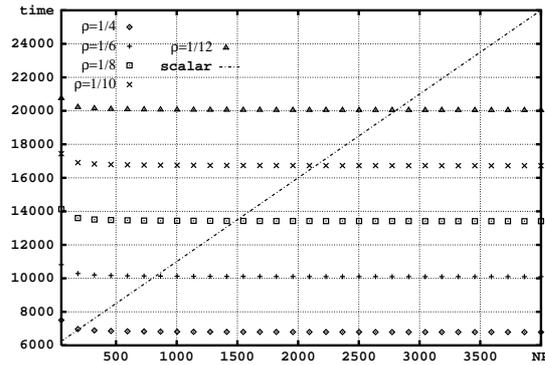

Figure 4: *Processing time as a function of the number of particles for different particle densities compared to the neighborhood–list method (dashed line).*

4. MD-Simulation of granular material using the new algorithm

The algorithm described above was intended to simulate the flow of granular materials like dry sand. Moving sand reveals very astonishing effects. When fluidized by shaking, it can behave like a fluid, while at rest it mostly behaves like a solid. Many experiments and computer simulations of moving sand have been performed.[8]

Using the proposed algorithm we investigated the flow of granular material

through a vertical long narrow pipe with diameter d and length l, $(l \gg d)$ and periodic boundary conditions. We want to describe our numerical simulations here only as an example to explain how the algorithm works and do not want to discuss the numerical results in detail which can be found in Ref. 9. To simulate a rough surface, the pipe was built of slightly smaller particles which interact with the freely moving grains in the same manner as the freely moving grains act on each other (fig. 5). For this reason we need not distinguish between the interaction of
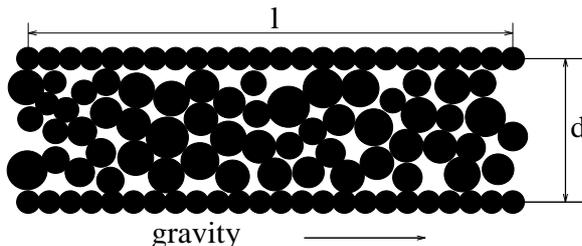

Figure 5: *The narrow pipe has a diameter d where only a few particles are allowed to fit. Its surface consists of slightly smaller particles to simulate a rough surface.*

the grains with each other and between the grains and the wall. The particle radii R_i have been chosen from a GAUSSian distribution with mean value R_0. For the force we assumed the Ansatz given by Cundall and Strack[10] and slightly modified by Haff[11]:

$$\vec{F}_{ij} = \begin{cases} F_N \cdot \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|} + F_S \cdot \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \cdot \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|} & \text{if } |\vec{r}_i - \vec{r}_j| < R_i + R_j \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

with

$$F_N = k_N \cdot (R_i + R_j - |\vec{r}_i - \vec{r}_j|)^{1.5} + \gamma_N \cdot M_{eff} \cdot (\dot{\vec{r}}_i - \dot{\vec{r}}_j) \tag{14}$$

and

$$F_S = \min\{-\gamma_S \cdot M_{eff} \cdot v_{rel} , \ \mu \cdot |F_N|\}, \tag{15}$$

where

$$v_{rel} = (\dot{\vec{r}}_i - \dot{\vec{r}}_j) + R_i \cdot \dot{\Omega}_i - R_j \cdot \dot{\Omega}_j) \tag{16}$$

$$M_{eff} = \frac{M_i \cdot M_j}{M_i + M_j}. \tag{17}$$

The terms $\vec{r}_i$, $\dot{\vec{r}}_i$, $\dot{\Omega}_i$ and M_i denote the current position, velocity, angular velocity and mass of the i–th particle. The model includes an elastic restoration force which corresponds to the microscopic assumption that the particles can slightly deform each other. In order to mimic three–dimensional behaviour the HERTZian contact force[12] which increases with the power $\frac{3}{2}$ was applied. The other terms describe the energy dissipation of the system due to collisions between particles according

to normal and shear friction. The parameters γ_N and γ_S stand for the normal and shear friction coefficients. Eq. 15 takes into account that the particles will not transfer rotational energy but slide on top of each other if the relative velocity at the contact point exceeds a certain value which depends on the normal component of the force acting between the particles (COULOMB relation[13]). For the parameters we have chosen $\gamma_N = 1000\ s^{-1}$, $\gamma_S = 300\ s^{-1}$, $k_N = 100\ N/m^{1.5}$ and $\mu = 0.5$. The system showed accurate numerical behaviour when the integration time step $\delta t = 10^{-4}\ s$ was used.

After starting the simulation with a homogeneous particle distribution, one finds the surprising result that the particles begin to form regions of different density which can move in both directions. These density waves are of no definite wavelength. The distances between the density maxima vary irregularly with time and depend strongly on the initial conditions. Figure 6 shows the evolution of the pipe. The pipe is drawn every 500 time steps, and the initial conditions are shown at the bottom. After detecting the density waves in our numerical simulations they were also found experimentally. For details see Ref. 9.
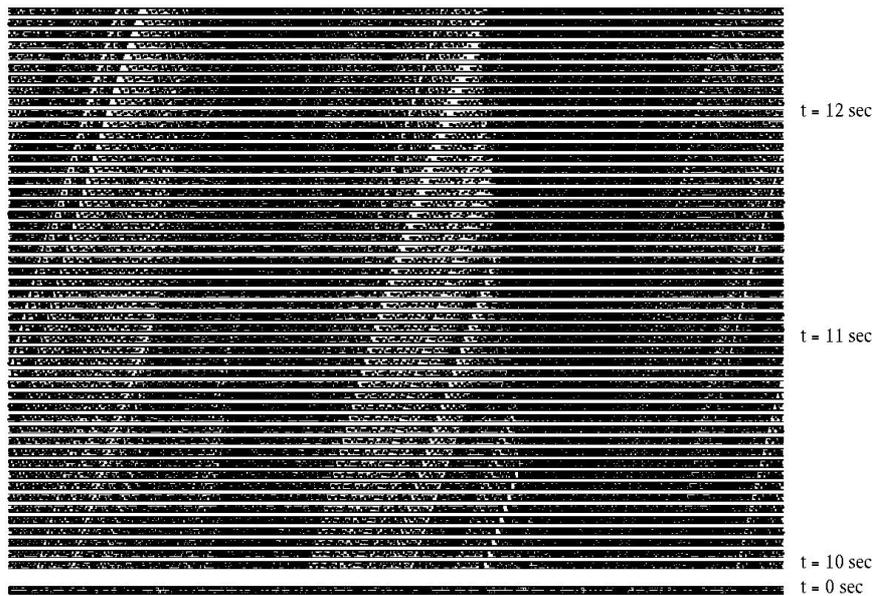

Figure 6: *The simulated flow of granular material through a pipe is shown in a sequence of snapshots. The snapshot on the bottom (t=0) shows the homogeneous particle distribution at the beginning of the simulation. The pipe is plotted every 0.05 seconds starting at time t=10 sec. The density profile varies irregularly with time and depends essentially on the initial conditions. Gravity acts from left to right.*

5. Conclusion

In conclusion we state that there is a lattice based algorithm of complexity $O(N)$ which can be vectorized completely and which acts very effectively given certain preconditions, such as high particle density, low dispersion of the particle size and short range interactions between the particles. The algorithm was compared in detail with the neighborhood–list method. For the case of higher particle dispersion, a similar algorithm can be defined using a mask of size 7×7 instead of 5×5. The algorithm was applied in a molecular dynamics simulation of granular material, and there were no differences in the results compared to equivalent simulations using the classical neighborhood–list method.

Acknowledgments

References

1. V. I. Arnold, *Mathematical Methods of Classical Mechanics*, Springer (New–York, 1978).
2. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes* (Cambridge, 1986).
3. M. P. Allen and D. J. Tildesley, *Computer Simulations of Liquids*, Clarendon Press (Oxford 1987); C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Englewood (New–York, 1971).
4. L. Verlet, *Phys. Rev.* **159**, 98 (1967).
5. G. S. Grest, B. Dünweg, K. Kremer, *Computer Physics Communications* **55** 269 (1989).
6. J. Barnes, P. Hut, *Nature* **324**, 446 (1986); L. Greengard, V. Rokhlin, *J. Comp. Phys.* **73**, 325 (1987).
7. F. Zhao, *Report AI–TR–995* MIT (1987).
8. H. M. Jaeger, C. Liu, and S. Nagel, *Phys. Rev. Lett.* **62**, 40 (1989); J. C. Williams, *Powder Techn.* **15**, 245 (1976); A. Rosato, K. J. Strandburg, F. Prinz, and R. H. Swendsen, *Phys. Rev. Lett.*, **58**, 1038 (1987); J. A. C. Gallas, H. J. Herrmann, and S. Sokołowski, *Phys. Rev. Lett.*, **69**, 1371 (1992); S. Sokołowski and H. J. Herrmann *Europhys. Lett.* **18**, 415 (1992); Y. H. Taguchi, *Phys. Rev. Lett.* **69**, 1367 (1992); G. W. Baxter, R. P. Behringer, T. Fagert, and G. A. Johnson, *Phys. Rev. Lett* **62**, 2825 (1989); T. G. Drake, *J. Geophys. Research* **95**, 8681 (1990); S. B. Savage and K. Hutter, *J. Fluid Mech.* **199**, 177 (1989); T. Pöschel, *J. Phys. (Paris)* **4** (1993).
9. T. Pöschel, preprint HLRZ–67/92.
10. P. Cundall and O. D. L. Strack, *Géotechnique* **29**, 47 (1979).
11. P. K. Haff and B. T. Werner, *Powder Technol.* **48**, 239 (1986).
12. L. D. Landau and E. M. Lifschitz, *Elastizitätstheorie*, Akademie–Verlag (Berlin 1989).
13. C. A. Coulomb, *Acam. R. Sci. Mem. Math. Phys. par Divers Savants* **7**, 343 (1773).

Appendix A

The algorithm of a GEAR predictor–corrector method reads as follows:

1. Predict the positions of the particles $\vec{r}_i^{\,pr}(t + \delta t)$ and the time derivatives $\dot{\vec{r}}_i^{\,pr}(t + \delta t)$, $\ddot{\vec{r}}_i^{\,pr}(t + \delta t)$, $\cdots$ up to the desired order of accuracy by TAYLOR expansion using the known values at time t:

$$\vec{r}_i^{\,pr}(t + \delta t) = \vec{r}_i(t) + \delta t \cdot \dot{\vec{r}}_i(t) + \tfrac{1}{2}\delta t^2 \cdot \ddot{\vec{r}}_i(t) + \tfrac{1}{6}\delta t^3 \cdot \vec{r}_i^{\,(3)}(t) + \cdots$$

$$\dot{\vec{r}}_i^{\,pr}(t + \delta t) = \dot{\vec{r}}_i(t) + \delta t \cdot \ddot{\vec{r}}_i(t) + \tfrac{1}{2}\delta t^2 \cdot \vec{r}_i^{\,(3)}(t) + \cdots$$

$$\ddot{\vec{r}}_i^{\,pr}(t + \delta t) = \ddot{\vec{r}}_i(t) + \delta t \cdot \vec{r}_i^{\,(3)}(t) + \cdots$$

$$\vdots$$

2. Calculate the accelerations $\ddot{\vec{r}}_i^{\,corr}(t + \delta t)$ acting upon the particles using the predicted values.

3. Correct the predicted values using $\ddot{\vec{r}}_i^{\,corr}(t + \delta t)$:

$$\begin{pmatrix} \vec{r}_i^{\,corr}(t + \delta t) \\ \dot{\vec{r}}_i^{\,corr}(t + \delta t) \\ \ddot{\vec{r}}_i^{\,corr}(t + \delta t) \\ \vdots \end{pmatrix} = \begin{pmatrix} \vec{r}_i^{\,pr}(t + \delta t) \\ \dot{\vec{r}}_i^{\,pr}(t + \delta t) \\ \ddot{\vec{r}}_i^{\,pr}(t + \delta t) \\ \vdots \end{pmatrix} + \begin{pmatrix} c_0 \\ c_1 \\ 1 \\ \vdots \end{pmatrix} \ddot{\vec{r}}_i^{\,corr}(t + \delta t)$$

where c_i are constant values depending on the desired order of accuracy.[3]

4. Proceed with the first step with incremented time $t := t + \delta t$.