IMD – A Massively Parallel Molecular Dynamics Package for Classical Simulations in Condensed Matter Physics

Johannes Roth, Jörg Stadler, Marco Brunelli, Dietmar Bunz, Franz Gähler, Jutta Hahn, Martin Hohl, Christof Horn, Jutta Kaiser, Ralf Mikulla, Gunther Schaaf, Joachim Stelzer, and Hans-Rainer Trebin

Institut für Theoretische und Angewandte Physik, Universität Stuttgart, Pfaffenwaldring 57, 70550 Stuttgart, Germany

Abstract. We describe the current development status of IMD (**I**TAP **M**olecular **D**ynamics), a software package for classical molecular dynamics simulations on massively-parallel computers. IMD is a general purpose program which can be used for all kinds of two -and three-dimensional studies in condensed matter physics, in addition to the usual MD features it contains a number of special routines for simulation of mechanical properties of solids, analysis and visualization.

1 Introduction

Molecular dynamics is a method widely used in physics and chemistry to study mechanical stability and thermodynamical properties of all kinds of systems from atoms to galaxies. Starting from the basic equations of motion which are ordinary differential equations with a huge number of variables (the positions and momenta of the particles) one derives a set of difference equations for finite time steps and solves the equations iteratively. In the equilibrium case one extracts thermodynamical averages like temperature, pressure or diffusion constants after the simulation has been finished. But MD simulations are not limited to this possibility: non-equilibrium simulations like heating, cooling, deformation of the sample can also be carried out.

The basic design of IMD is due to Jörg Stadler [1] who wrote most of the code as a contribution to the project A2: "Molecular Dynamics Simulations of Quasicrystals" of the collaborative research center (Sonderforschungsbereich) 382: "Procedures and Algorithms for the Simulation of Physical Processes on Supercomputers". Meanwhile IMD has been enhanced by many new features. The software is currently maintained by a team of programmers. To organize the ongoing development and maintenance we use the software management system CVS. Although we intend to give the most up-to-date description of IMD in this article, we would also like to draw the attention of the reader to the IMD-webpage [2] where the latest news, the source code and the documentation can be found. The design and development of IMD has been described in two articles, called Paper I [1] and Paper II [3]. An additional HLRS report will present some of the results achieved with IMD [4].

1.1 Short overview of IMD 1.3

Interactions and Ensembles. IMD 1.3 allows particles to interact via pair, three-body [4] and EAM[1] potentials. Details of the implementation are given in the second report. Further interactions are the Gay-Berne potentials commonly applied in liquid crystal studies. These interactions are anisotropic modifications of the well-known Lennard-Jones potentials [6]. There is no limit to the number of particle types differing in mass and interaction potential. The pair interaction potentials are read from tables and therefore are not limited to simple analytical expressions. The only exception are Lennard-Jones-potentials where the interaction is computed directly if the system is monatomic. The calculations can be carried out in a number of thermodynamical ensembles [7] like microcanonical (NVE, total energy is constant), canonical (NVT, temperature and volume constant, NPT, temperature and pressure constant, and NσT, temperature and uniaxial stress constant). For NVT a stochastic Andersen thermostat [7] is also available. In general a Nosé-Hoover thermostat [7] is used for NVT, NPT and NσT. In addition to MD simulations it is possible to optimize the potential energy of a sample with the microconvergence method [8]. Mechanical boundary conditions allow the sample to be pulled apart or sheared at constant rates or certain intervals.

Multi-Phase Simulations. In very large simulations it may not be possible to read and write the initial and final data sets. On the other hand it is often necessary to optimize the potential energy of the generated structure with the microconvergence method first, then to equilibrate it with NPT to correct the volume, and finally to carry out an NVT or NVE simulation to collect the data for the thermodynamical averages. With IMD it is possible to carry out such a multi-phase simulation in a single run.

Simulation Setup. Several types of crystal or quasicrystal structures may be generated directly with IMD. In addition there are special setups for fracture simulations to damp out shock waves emitted from a crack [9], and in shock wave simulations to initiate a shock wave. A moving crack generates heat which must be removed from the sample (usually this is done by long range heat conduction). Since the ordinary Nosé-Hoover thermostat mimics an infinitely large thermal conductivity we had to modify the simulation setup: in the central part of the sample a NVE ensemble is simulated, and the Nosé-Hoover thermostat is limited to the outer part of the box.

Online Tools. A common usage of MD simulations is the calculation of materials properties. For this purpose special functions or the collection of histograms of several variables have to be evaluated. This can be done effectively only during simulation. IMD contains routines to compute self-correlation

[1] EAM: **E**mbedded **A**tom **M**ethod, first described by Daw and Baskes [5].

functions which are used to study the dynamical properties of solids. Transport properties like the thermal conductivity for example can also be computed. To monitor atomic jumps of atoms we have implemented flip detectors which allow to keep track of the motion of pre-selected atoms. The flip detectors can be applied only to certain decagonal quasicrystals since the possible jumps depend heavily on the geometry of the simulated structure.

To study the dynamics of a simulation it is helpful to know how the kinetic and potential energy changes. The kinetic energy for example allows to locate the crack tip whereas the potential energy is sensitive to stacking faults. With IMD one can produce color-coded histograms of both energy distributions.

It is also possible to compute displacement fields and energy difference maps if reference states are stored at the beginning of a simulation.

Online Visualization. Large simulations on massively-parallel supercomputers require completely new concepts to deal with the overwhelming flood of data: If there are ten to hundred million particles and the simulation is carried out for several hundred thousand or million steps it is impossible to store the complete particle configurations, even at infrequent time intervals. IMD includes several features to deal with this problem: It is possible to generate pictures of the potential or kinetic energy directly. The program also provides the possibility of online visualization: the simulation on a remote supercomputer is monitored at the same time on a local workstation (even if there is a firewall in between as it is the case at the HLRS).

Metacomputing. The distribution of one application on a cluster of parallel computers is called metacomputing. It is possible to carry out such simulations with IMD.

Implementation Details. The package's main design goals were to create flexible and modular software that reaches a high performance on contemporary computer architectures while it is still as portable as reasonably possible. The target platforms for IMD are RISC workstations and massively-parallel computers with distributed memory. It does not perform well on vector machines since the vectorizable loops are to short. To run IMD efficiently on a distributed-shared architecture would also require some recoding.

IMD is actively supported on DEC and SGI workstations and on PCs running Linux as well as on the Cray T3E. It has been ported to IBM workstations, to the Intel Paragon and to the Hitachi SR2201. IMD is written in C and uses the widely available MPI library for message passing. It has been adapted to metacomputing via the PACX [10] library. For the online visualization a special version of VolRend [11] called VolIMD is available.

Offline Tools. IMD is not only a simple simulation program but a package which contains routines that allow to analyze and visualize the results. This means that their are tools which share the data structures set up in IMD.

The analysis tools usually start from configuration files which contain the positions and velocities of the atoms, their potential and kinetic energies, their type and masses and further properties if required. There is a program which computes the radial distribution functions for each type of atoms from the coordinates. PPM pictures or VRML scenes can also be generated.

Outline. In Paper I the concept of linked cells for simulations of short-range interactions and its the extension to parallel computers has been described in detail. Paper I also deals with communication modes and the performance of IMD in large scale and high speed simulations. In Paper II we have described the concepts of online visualization and metacomputing in detail.

This paper is organized as follows: in the first section we will summarize Paper I and extend it to the concept of dynamical linked cells and many-body interaction. In the second part we will describe the online visualization and the generation of pictures. In the third section we will talk about meta-computing and In the final section we present examples and summarize the applications that have been simulated with IMD.

2 Basic Concepts of Short Range Massively-Parallel MD Simulations

In MD simulations the equations of motion for a system of N particles have to be solved over a time T_{sim}. The integration is done stepwise in small time intervals Δt. At each step the force F_i acting on particle i must be calculated. Since the particle may interact with all other particles, the complexity of the problem grows as $O(N^2)$. Thus the force calculation is the most time consuming part of the simulation. If the forces are short ranged, however, the complexity for the necessary force calculations grows with NN_c, where N_c is the number of atoms in the range of the interactions. This is true for two-body pair potentials as well as for many-body interactions like EAM for example, since the many-body interactions are usually also short ranged.

For short-range interactions the computation of forces is a local problem and lends itself naturally to parallelization by domain decomposition. The simulation box is geometrically divided into cells which represents an easy way to find the neighbors of any particle.

In case of pair interactions the total force F_i acting on particle i is the sum over all forces f_{ij} of the particles j within the interaction range r_c. To calculate F_i we must know the positions r_j of all particles closer than r_c to particle i. To find them we subdivide the simulation box into small cells with a diameter that is at least twice as long as r_c (see Fig. 1). The cells need not be squares or cubes, IMD also permits parallelepipeds. The atoms can easily be assigned to the cells if their coordinates are divided by the length of the simulation box and then multiplied by the number of boxes. Since the diameter of the cells is larger than $2r_c$, all interactions take place within a cell or between neighboring cells. What remains to be done is to set up a linked list of the neighbors of each cell once at the beginning of the simulation. Then the whole computation can be carried out by computing the interaction between atoms in the same cell and between atoms in neighboring cells. Since $f_{ij} = -f_{ij}$ by Newton's law only half the number of neighbors has actually to be taken into account (See Fig. 1). Since the assignment of the atoms to the cells is a linear process the complexity of our simulation has been reduced to $O(N)$.

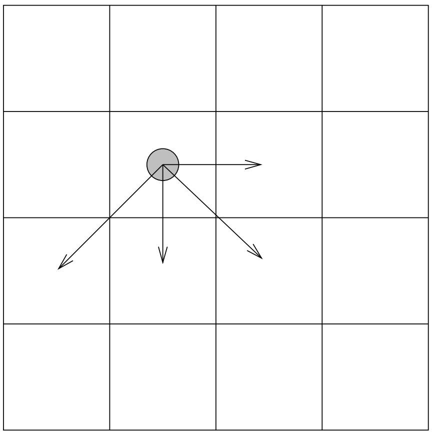

Fig. 1. Decomposition of the simulation box into cells. The arrows indicate the neighbor cells that must be taken into account in the computation of the interaction in the central cell.

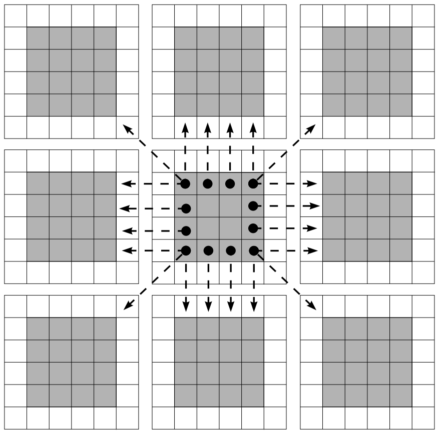

Fig. 2. Assignment of the cells to processors in two dimensions. The white cells are *buffer cells*, the adjacent cells (dotted in the central processor) are the *surface cells*.

In the NVE and NVT simulations the cell structure is set up at the beginning of the simulation and remains unchanged during the whole run. This is not the case in NPT or NσT simulations where the cells must be adjusted to the variable volume. It may happen that the diameter of the cells is becoming too small if the volume shrinks for example during a transition from a gas to a liquid. Or during the opposite transition it may happen that the number of cells is too small and the simulation becomes very inefficient. In NPT and NσT simulations IMD checks if the cells decomposition is appropriate, and if not, it computes a new cell setup.

2.1 Cells on Parallel Computers

The generalization of the cell structure to parallel computers is rather straightforward: IMD assumes that the processors are arranged on a two- or three-dimensional cartesian grid with periodic boundaries, so each processor has 26 neighbors in three dimensions or 8 in two dimensions. Our algorithm subdivides the array of cells into equally sized parts which then are assigned to the processors. Figure 2 shows a two-dimensional example of how the cells are associated with the processors: A 12×12 array of cells (shaded) is distributed onto a 3×3 grid of processors. A layer of empty cells (non-shaded) is added to each processor. These are used as temporary stage for particle data received from neighbors. We call the extra cells *buffer* cells. The outermost layer of non-empty cells at the surface of a processor's part are called *surface* cells.

Particles in non-surface cells interact only with particles on the same processor, whereas those in the surface cells also interact with particles in surface cells on the neighboring processors. To calculate the forces for them, data from the surface cells are copied to the buffer cells of the neighbors, as indicated by the arrows in Fig. 2.

Three communication steps have to be performed at each time step: First, the particle positions have to be exchanged with the neighbor processors for the force calculation, second, the forces accumulated due to Newton's third law by particles in the buffer cells are sent back to the original processor, and third, after the positions have been updated, all atoms that have left or entered a processor's column must be transferred to their new processor.

As indicated in Fig. 2 by the arrows, the data for each surface cell could be communicated cell by cell. This is, however, not optimal since it creates a large number of small messages. IMD first collects the data for a given neighbor into an intermediate data buffer and then translates the whole buffer with only one single MPI call. We have tested the cell by cell and the buffer scheme. Details are given in Paper I.

In IMD periodic boundary conditions are usually assumed in all directions, but it is possible to switch them off and use open or fixed boundaries. If the system is homogeneous and periodic boundary conditions are used, we find that load balancing conditions are fulfilled very well. However, if open or fixed boundaries are present or inhomogeneous parts like cracks or shock fronts we find that the load balancing goes down considerably. The minimum was a computational effectivity of 35%. An improvement of this situation would require the introduction of active load balancing.

3 Online Visualization and Picture Generation

Until recently MD simulations were carried out in the following way: The simulation was run, data were collected and stored for later analysis. With the advent of large massively-parallel supercomputers such a scheme may no longer be feasible since in many cases the amount of data is too large to be stored.

One way to circumvent this problem is to visualize the data online. More advanced schemes even permit to steer the simulation online, for example by changing the applied pressure or temperature, by altering the frequency of picture production or by starting data storage or interrupting the simulation in case of errors.

With IMD online visualization is possible and the basic interfaces for online steering are also given. IMD uses ordinary TCP sockets for communication. This may cause problems if the supercomputer is hidden behind a firewall as it is the case at the HLRS. For this reason IMD works with high port numbers, but it may still be necessary to ask the computing center to open up ports for communication. More details about the interaction of simulation and visualization can be found in Paper II.

Online visualization can be used for interactive jobs as well as for batch jobs. The monitoring may be started at any time during the simulation. Then the visualization client is connected to the simulation server and receives pictures or histograms. If the simulation is running for a long time, one may connect to the server infrequently and check the status of the simulation job.

For the online visualization a program is required which runs on the visualization client and is able to deal with the IMD output. This program is called VolIMD, a modification of VolRend written by R. Niemaier [11]. Although VolIMD is running only on SGI workstations, there exists a version of VolRend called GlutRend which can be used on DEC Alpha workstations and on Linux PCs. All the programs are available at our institute.

3.1 Pictures and Movies

The pictures and histograms produced by the online visualization routines are raw data written in binary format. This has the advantage of being faster and more flexible than writing formatted output. At the beginning of the simulation one has to define an array of bins into which the data are collected. The conversion of the binary data is carried out by the ppm-tools or with the help of VolRend. In two dimensions it is possible to write ppm-files directly from IMD, but then one looses the possibility to manipulate the pictures afterwards.

Picture sequences from the simulation, from VolRend or from the offline tools may be assembled to computer movies using movieconvert from SGI for example or any other movie program. This adds a new dimension to the presentation of the results since the dynamics of a system can be studied directly.

4 Special Requirements for Metacomputing

IMD has been written for massively-parallel supercomputers like the Cray T3E, where the communication between the processors is very fast and and the communication time is short compared to the time required for a simulation step. For metacomputing this is no longer the case. The internal communication time within the T3E is 18 μs, with PACX it is 21 μs. Communication between the supercomputer nodes is at least one order of magnitude slower than within a supercomputer, even if high-speed and high-bandwidth lines are used. The latency time to send a the message between two partitions within the same T3E is 3.5 ms, between Stuttgart and Manchester it is 45 ms, and between Stuttgart and Pittsburgh it is 75 ms! To avoid wrong results the computations have to be stopped until the inter-supercomputer communication has been finished. This is very unsatisfactory and a way to interlace calculation and communication has to be found: If we take a look at the central part of the middle cell in Fig. 2 we find four gray cells without dots. The calculations in these cells can be started at the same time as the communication. Then we may proceed with the surface and boundary cells which have been updated already, thus breaking up the synchronous communication scheme into smaller pieces which can be sent continuously during computation.

5 Applications

Up to now, IMD has been used mainly for simulations of the mechanical stability of quasicrystals. Examples are simulations of cracks, fracture [13–15], shear deformations and dislocations in quasicrystals in two and three dimensions [16], shock waves in crystals and quasicrystals in three dimensions [3,12], interfaces and covalent materials. Details of the latter results are given in a second HLRS report [4]. Upcoming simulations include the study of copper islands on gold under irradiation, the simulation of silicon interfaces and the modeling of milling processes. Here we give a short summary of two applications. Details may be found in the original publications. For a typical run the simulation box contains about 70000 atoms. On the Cray T3E it runs on 64 processors and lasts about 30 minutes. This permits us to run parameter studies at various loads and temperatures.

Crack Simulations. For the crack simulation we use a piece of quasicrystal with fixed boundaries at the bottom and at the top and free boundaries at the other two sides. The material in a notch is removed from the sample, and then the upper and lower boundaries are moved apart by a fixed amount until a crack is emitted from the notch. The breaking of bonds at the crack tip bonds causes the emission of shock waves. To avoid an interaction of the crack with the shock waves reflected at the boundary of the sample, we have generated a stadium around the crack. Waves meeting the stadium are damped out [9]. In a crystal the crack would propagate smoothly. In a quasicrystals the crack is meandering around, thereby avoiding clusters of strongly bound atoms (Fig. 3). The motion of the crack has been studied at various loads: it starts from an intermittent behavior at a small load and gets continuous at higher load. The velocity increases and saturates. At high loads the crack motion gets chaotic since the energy released cannot be transported away fast enough.

Shear Simulations. Further studies are concerned with shear simulations of quasicrystals. In this case the upper border of the cell is shifted parallel to the lower border but in the opposite direction. Dislocations are created which move through the quasicrystal. In contrast to the crystal case the sample is not perfect behind a dislocation but contains a stacking-fault-like phason wall (Fig. 4). On the one hand this weakens the quasicrystal, on the other hand it costs energy. Dislocation pairs are created on the defect wall which modify it in such a way that the energy cost is reduced.

Acknowledgments

The help of Michael Reichenstein (visualization) and the PACX group led by Michael Resch and Edgar Gabriel is gratefully acknowledged.

References

1. Stadler, J., Mikulla, R., Trebin, H.-R.: IMD: A Software package for molecular dynamics studies on parallel computers. Int. J. Mod. Phys. C **8** (1997) 1131-1140
2. www.itap.physik.uni-stuttgart.de/~johannes/imd-home.html
3. Roth, J.: IMD – A molecular dynamics program and applications. Proceedings of the workshop: MD on Parallel Computers. Eds. Attig, N., Esser, R., to be published by World Scientific, Singapore
4. Hahn, J., Trebin, H.-R.: Molecular dynamics of covalent systems, HLRS report 1999, this volume
5. Daw M.S., Baskes M.T.: Embedded atom method: Derivation and application to impurities, surfaces and other defects in metals. Phys. Rev. B **29** (1984) 6444-7991
6. Gay G.J., Berne B.J.: Modification of the overlap potential to mimic a linear site-site potential. J. Chem. Phys. **74** (1981) 3316-3319
7. Allen M.P., Tildesley D.J.: Computer simulation of liquids. Oxford Science Publications, Oxford (1987)
8. Beeler J.R.: Radiation effects computer simulations. North Holland, Amsterdam (1983) p 27
9. Zhou S.J., Lohmdahl P.S., Holian B.L.: Dynamic crack processes via molecular dynamics. Phys. Rev. Lett. **76** (1996) 2318-2321
10. www.hlrs.de/structure/organisation/par/projects/pacx-mpi/
11. www.uni-stuttgart.de/RUSuser/vis/People/roland/volrend/volrend.html
12. www.hlrs.de/news/events/1998/sc98/sc98_hlrs_booth.html
13. Mikulla R., Stadler J., Krul F., Trebin H.-R.: Gumbsch P.: Crack Propagation in Quasicrystals. Phys. Rev. Lett. **81** (1998) 3163-3166
14. Mikulla R., Stadler J., Gumbsch P., Trebin H.-R.: Molecular dynamics simulations of crack propagation in quasicrystals. Proceedings of the 6th International Conference on Quasicrystals. Eds. Takeuchi S., Fujiwara T., World Scientific, Singapore (1998) 485
15. Mikulla R., Gumbsch P., Trebin H.-R.: Dislocations in quasicrystals and their interaction with cluster-like obstacles, to appear in Phil. Mag. Lett.
16. Schaaf, G.: Numerische Simulationen des dynamischen Verhaltens von Quasikristallen. Diploma Thesis, Stuttgart (1998)

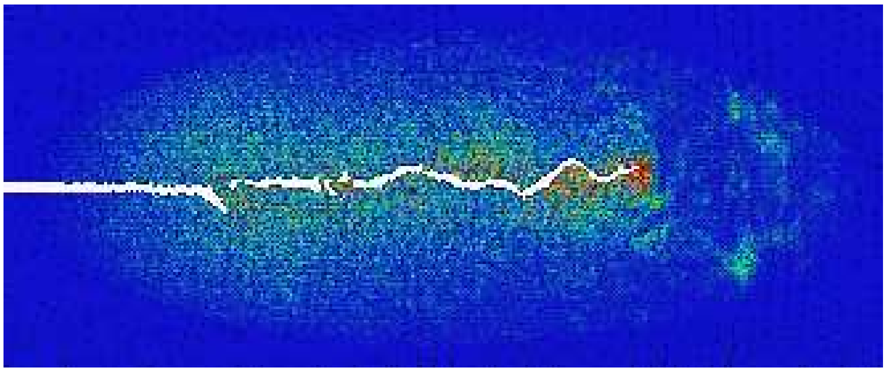

Fig. 3. Crack in a quasicrystal. Blue indicates low energies, green the shock waves emitted from the crack and red the very hot crack tip.

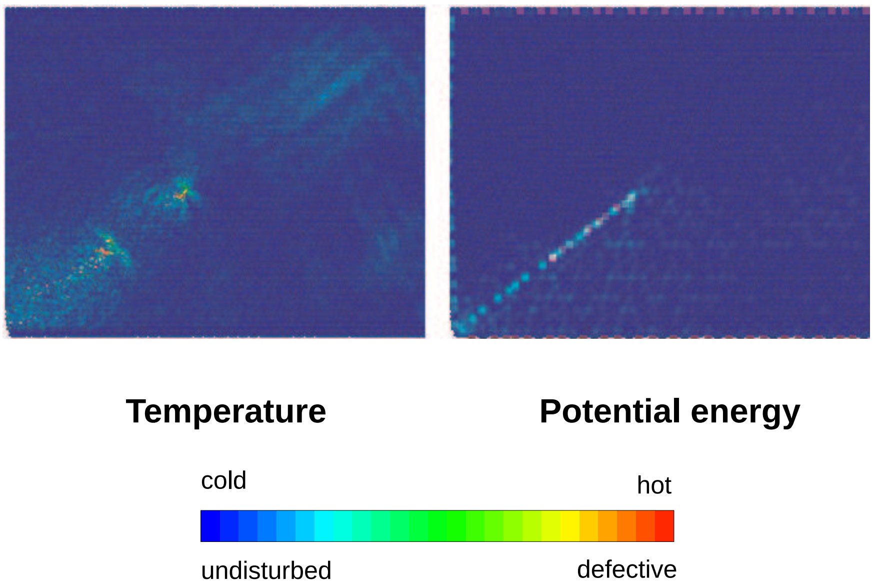

Fig. 4. Shear simulations of quasicrystals. Left side: temperature of the sample. The green parts indicate the position of the dislocation defects. Right side: the green chain of dots marks the location of the stacking fault.