A Tool for Materials Exploration

Dieter W. Heermann
Andreas Linke
Christian Münkel

Institut für Theoretische Physik
Universität Heidelberg
Philosophenweg 19
69120 Heidelberg
and
Science in Software
An der Münzenbach 4
69151 Neckargemünd

Abstract

We present a tool for the simulation and analysis of materials properties. This tool comprises simulation methods for models of materials (metal and polymer materials) as well as visualisation techniques to explore materials. In this paper we describe the basic design principles.

1. Introduction

The complex demands on materials require simulation tools to rapidly design new or redesign old materials with prescribed properties. The complexity and richness of materials make it impossible to have just one generic tool comprising all situations. It is possible to design simulation methods that in principle are able to handle the simulation of models for the design of materials. Examples are the Monte Carlo and the molecular dynamics method, as well as methods derived from these two methods. In practice there are insurmountable problems if one tries to apply the bare or generic methods. In most cases a modification of the basic simulation method and/or the model is necessary to surmount the difficulties. Unfortunately most of the simulation tools and packages are closed in the sense that the internal interfaces and calling sequences are neither known nor available to the end user. This situation is somewhat unsatisfactory for a user of simulation tools who must adopt the methods or the

models to the problem at hand. The situation has changed slightly in favour of a more open approach recently. Here we describe a package for the simulation of materials properties that encompasses the openness as one of the main design goals. The End-User can write own simulation routines and incorporate them into the tool. The tool supplies the user with a framework that has the look and feel of graphical user interfaces commonly in use today in the industrial and academic research and development workplace. It is entirely based on the industry standard X-Window system making it possible to use existing computing facilities. No special graphics acceleration is necessary though supported if available on the host machine. Furthermore the algorithms make use of inherent parallelism in the simulation methods and models. This makes possible to run the tool in a parallel environment may it be coupled workstations, a symmetric multi-processing workstation or mainframe machines. Even at this point the user can link into the package since the MPI message passing standard is supported by the package. Such a package requires the cooperation between industry, university and software house. Also many problems require further research in the methods and models before they lead to a routine application in the industry laboratory.

2. Design Principles

The Materials Explorer (mxpl) provides a framework for simulations of materials properties. It is not meant to be a universal tool covering all situations or providing modules designed to perform specific tasks. The main design principles are to supply a basic functionality, with the characteristic look and feel of graphical user interfaces, open for user written modules.

The first step for the simulation of materials properties is the modelling. The modelling dictates often the kind of simulation method. If a standard model is applicable, then also a standard simulation method and algorithm. mxpl provides a standard set of models like lattice models for the simulation of metals or off lattice continuum models for the simulation of macromolecular materials. Corresponding to these models mxpl makes available simulation methods.

In this situtation as well as in cases requiring a new model and/or method the production cycle is split into

- preprocessing
- simulation (production runs)
- postprocessing

Pre-processing prepares the input for the models. In the framework of materials simulation the input is a configuration of atoms, molecules, etc. This stage can already involve a simulation. The simulation or production run stage transforms the input into a sequence of output configurations.

The basic idea for the framework provided by mxpl is the *actor model*. Central to all simulations is the configuration. We view all stages as manipulating or acting on a configuration or sequence of configurations. The configuration is centre for all action. The user chooses a configuration. The configuration file carries information on the model or models it is meant for. From this information mxpl derives the possible pre,

postprocessing and simulation actions. A selection of a lattice configuration spins off simulation methods, for example Monte Carlo or Hybrid-Monte-Carlo simulation, and inhibits others, for example molecular dynamics. In the window for the methods only those applicable for a chosen configuration appear.

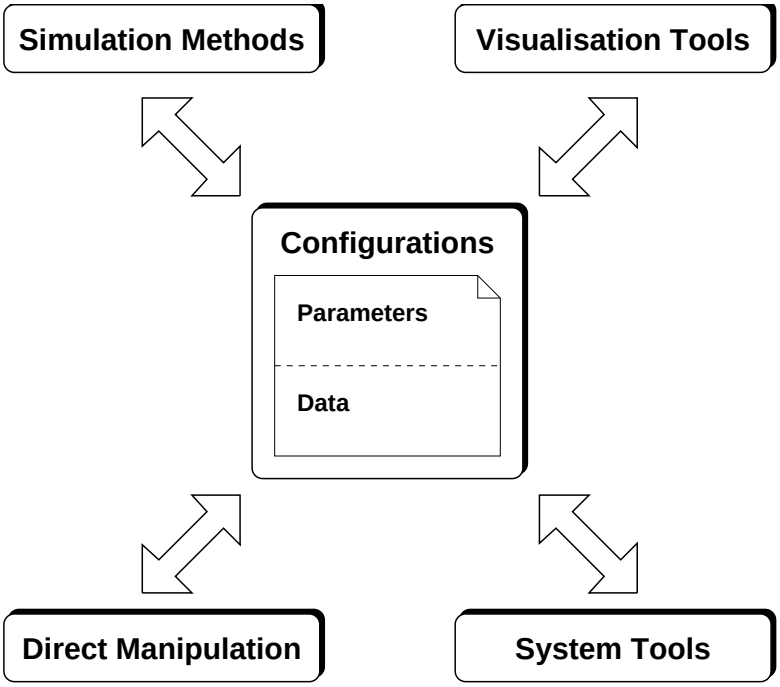

Figure 1: Actor model for the Materials Explorer mxpl. The configurations are at center upon which the actions are placed

In the pre- or post-processing stage the configuration can be manipulated by graphical tools. One example is the molecular force microscope. It allows to explore and manipulate a configuration by displacing monomers. The user can thus develop an intuition into the material since also forces are displayed indicating the action that will take place in a subsequent simulation.

Graphics is one of the design principles. The understanding of the properties of materials by simulation yielding materials constants is one aspect the other being to develop an intuition into what is necessary to change the properties. Looking and playing with the material can foster the intuition. Therefore strong emphasis is put on the graphical part of the tool. Both pre- and postprocessing should not rely on a special graphics engine. The user should be able to work remotely perhaps utilising a compute engine and displaying the results of simulations on the native platform. This requirement naturally leads to developing the graphic part using standard X Window calls. These calls are packaged in a library hiding most of the details from a user who needs to write his interface or graphical exploration routine. This part of the design principle ties in with the general directive of absolute openness. All interfaces are made public to make it possible to link in custom-made and customer-made programs.

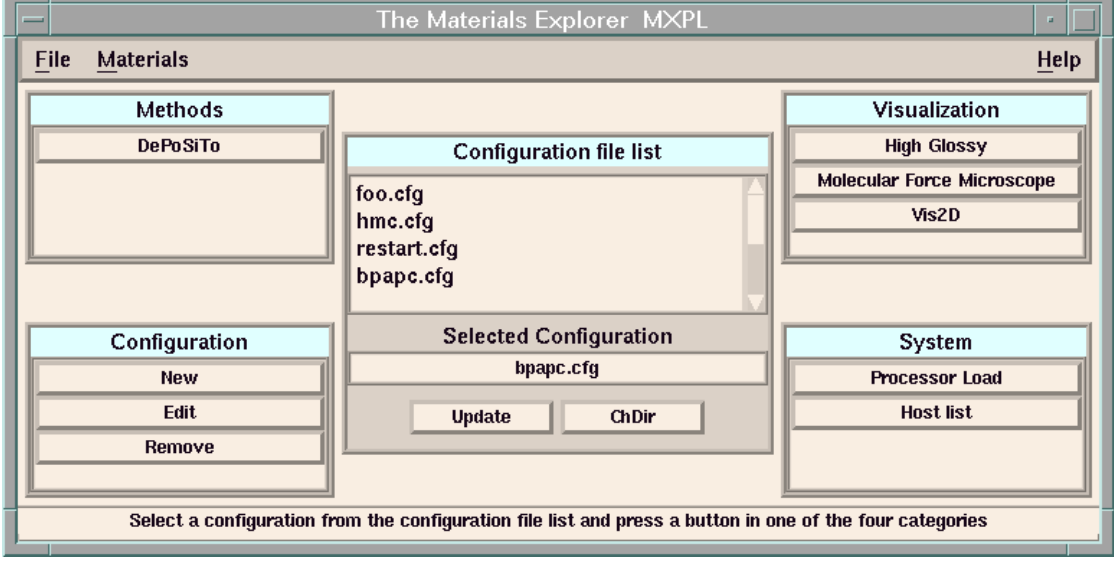

Figure 2: The figure shows an example of the primary window of mxpl

The last principle concerns the inherent parallelism in the models and the algorithms. Most of the simulations are time consuming. Simulations taking weeks to come up with one property of a material need to be cut down to facilitate rapid development. To exploit the inherent parallelism is evident. The design and choice of the algorithms for the simulation part was guided by the goal of making maximum use of existing hardware and software platforms.

2.1. Communication Model

Files are the means of communication between programs. Configuration files play the key role in all actions (see figure 1). Using files to transmit data is surely not the fastest way on a workstation (compared with e.g. UNIX sockets or rpc calls) but still is the most reliable and safest. With support for multiple platforms with different operating systems it may very well be the only way available.

Configuration files contain both the parameters for a simulation and the actual configuration data. This implies a two-folded structure of the configuration file with a standardized first part containing parameter keywords and values and a second application-dependend data part. For each configuration the proper parameters (describing e.g. the physical conditions of the simulation) are always available together with the actual data (e.g. the coordinates in configuration space) which makes bookkeeping a lot easier.

2.2. User interface

The main window of mxpl (figure 2) mirrors the principle model described in figure 1. There is a central file selection box where configuration files to work on can be selected. Depending on the selected configuration file, pushbuttons for starting the appropriate application appear in the four main application categories. These applications can be either built-in tools or stand-alone programs. The whole interface is open and transparent to the user and can be customized to the specific needs of a site.

2.2.1. External and internal applications

As extensibility and configurability are a main issue it should be easy for the user to tailor the user interface to specific needs. Extending and configurating the user interface means adding or removing application programs, i.e. simulation or analysis tools as well as utility programs.

There are two possible ways of adding applications to the user interface: Internal (linked) and external (standalone) applications.

Internal applications are linked into the executable mxpl. They start faster than external applications, because no searching for pathnames is needed and no system call overhead for starting the program is necessary. Internally linked applications may also benefit from the user interface convenience functions in mxpl, e.g. for finding initialization files in the proper places and parameter input. However, linking many applications into the main program significantly increases the size of the executable, making it slow to startup or even impossible to run on machines with little memory. Furthermore, for every new developed or updated program all applications and libraries must be linked together. This may be not only unconvenient but also requires system programming knowledge which can't be supposed of every user. Debugging of new programs is much easier for stand-alone applications. Therefore, to make flexible adding and removing of applications to the user interface possible, mxpl allows for incorporating external applications into the user interface at runtime. This is achieved by scanning a description file, describing the characteristics of all external applications (like the category and the required startup parameters). For each application, a parameter description file stating the type (e.g. integer, real or boolean value) and possible default values and bounds of parameters is scanned. The external applications description files are in plain ASCII text format and can be changed from within mxpl as well as with every text editor. This approach allows for flexible adding/removing of tools and applications into mxpl as needed. The interface between the application programs and mxpl is standardized through the use of libraries and header files for reading configuration files and for window-based input of parameters.

From the user's point of view, both internal and external applications are treated on the same footing: Depending on the selected configuration file, suitable applications appear as pushbuttons in the user interface (see figure2). Clicking on one of these buttons starts the application corresponding to the currently selected file as input. The application scans the input file for parameters and pops up a parameter input window to allow the user to change or adapt parameters. After finishing the simulation the output files may be analyzed and the results visualized.

2.2.2. Portability

To ensure compatibility with the vast majority of workstation platforms, the user interface is based on the de-facto standard X-Windows X11R5 (with the Motif 1.2 toolkit) for user interfaces on UNIX workstations. This graphical user interface is supported by all major Unix vendors. The visualization routines use X-Windows library calls and are thus not dependend on special rendering software and at the same time fast enough to provide acceptable speed even on low-end machines. The parallelization tools support the message passing interface standard MPI[5] which allows parallel programming on all major parallel computer systems as well as workstation clusters.

The Materials Explorer has been implemented and tested on all major Unix platforms including Sun Sparc10 with Solaris 2.3, IBM RS/6000 and SP/2 with AIX 3.2.5, SGI Indigo with Irix 5.2 and HP PA 735 with HPUX 9.1. Therefore even simulations in a heterogenous environment are supported. MPI is available in several public domain implementations for all of these platforms. Commercial (vendor) implementations of MPI also exist or are announced for some of these platforms.

3. The Simulation Methods

The tool contains some basic simulation methods. Once the user has chosen a problem, i.e. a configuration, then only certain methods are applicable. The system automatically selects the methods that make sense to apply to the configuration. Currently the generic methods are

- Monte Carlo Methods
- Molecular Dynamics Methods
- Stochastic Dynamics and
- Hybrid Dynamics.

The algorithms implementing the methods are not generic in the sense that they apply to all models. The algorithms, if they are to perform with good efficiency must be tailored to the model. For this reason the methods and algorithms in the basic set of the package apply to standard models for materials where good implementations exist. New models or variation of the existing model base requires a re-design of the algorithms. In the spirit of the design principle of the package this can be done by the end-user since the structure is open or can be done in collaboration.

A large class of many-particle systems can be described by a classical Hamiltonian of the form

$$H(x,p) = T(p) + U(x)$$

where $T(p)$ denotes the kinetic energy (p denotes the monenta) and $U(x)$ the potential energy with its configurational dependence made explicit by the variable x.

In the Monte Carlo method one calculates the expectation value for an observable A (for example the concentration, the magnetization, correlation function etc.) by computing the average with respect to an appropriate distribution. The distribution function is determined by the statistical mechanical ensemble. For the canonical ensemble with fixed temperature T, volume V and number of particles N the expectation value for an observable A one treats the configurational degrees of freedom such that one samples those states in the phase space of the system that contribute most to the averaging for the observables. One does so by changing the configurational degrees (for example displacement of atoms). This results in a change in the energy (H). If the energy is reduced by the move it will always be accepted. If the energy is raised, the move will be accepted with a probability propotional to the Boltzmann factor for the system, i.e., *exp{-Change in H/ Temperature}*.

In conventional Monte Carlo (MC) calculations of condensed matter systems, such as an N-particle system with a Hamiltonian $H = T + U$, only local moves (for example

displacement of a single monomers in a polymer system, single spins in magnetic systems) are made. Updating more than one particle typically results in a prohibitively low average acceptance probability for the new state of the system. This implies large relaxation times and high autocorrelations especially for macromolecular systems. In a Molecular Dynamics (MD) simulation, on the other hand, global moves are made. These result from the numerical integration of the equations of motion. These can be obtained from the newtonian, Lagrangian or Hamilton equation of motion. The MD scheme, however, is prone to errors and instabilities due to the finite step size in time for the numerical integration. In order to introduce temperature in the microcanonical context, isokinetic MD schemes are often used. However, they do not yield the canonical probability distribution, unlike Monte Carlo calculations.

The Hybrid Monte Carlo (HMC) method combines the advantages of Molecular Dynamics and Monte Carlo methods: it allows for global moves (which essentially consist in integrating the system through phase space); HMC is an exact method, i.e., the ensemble averages do not depend on the step size chosen; algorithms derived from the method do not suffer from numerical instabilities due to finite step size as MD algorithms do; and temperature is incorporated in the correct statistical mechanical sense.

It is thus essential to be able to try various methods on a particular system. Each may have specific advantages for the question one would like to answer. Indeed, often all of the above methods can be applied to one model (for example systems with continous dregrees of freedom!). All of the above mentioned methods have shown their merits on numerous occasions. An excellent review is given in reference [x].

The models for various materials problems so far implemented have shown their value for many problems. Here we only want to mention just a few in the references [x].

During the design we put special emphasis on the parallelization of the algorithms. Those susceptible to parallelization and promising good speed up potential on the generic models were selected and developed. To facilitate the handling of the parallel run environment the parallelization is transparent to the user.

4. Parallelization

There are three main approaches to scientific parallelization:

- trivial or poor man's parallelization, i.e. starting a serial program on various machines with different initial conditions
- data decomposition, i.e. distribution of data on different machines
- domain decomposition, i.e. distribution of configuration space on different machines.

All these parallelization schemes are applicable to workstation clusters (with low bandwidth interconnection) as well as true parallel machines (with shared or distributed memory).

Unlike data or domain decomposition, trivial parallelization has the advantage that no change of the serial program is necessary. Starting several identical programs with different initial conditions may improve the statistics of the results as well as cover a broader range of environmental parameters in the same time. There are, however, cases where the calculation time of one single run is crucial, especially when very large systems are investigated to minimize finite-size artefacts, or many iterations are necessary to cover a long time scale. Especially for very large systems with local interactions, domain decomposition turns out to be most efficient. In the simplest case, a rectangular lattice is divided into sub-lattices of equal size, and each processor updates one single sub-lattice. Because of the locality of the interactions, only a small boundary area of the sub-lattices must be communicated to neighbour processors during a single update. Therefore, for large lattices, the efficiency of the parallel program is close to the ideal value 1. The efficiency of a parallel program is measured by

$$e = T_{serial} / n\, T_{parallel}$$

where n is the number of processors, $T_{parallel}$ the time of the parallel program and T_{serial} the time needed for an (almost) identical serial program for the same problem.

mxpl supports the user in all three parallel programming paradigms. In the simplest case, a serial program will be started on different machines with varying initial conditions, e.g. different seeds for the random number generator or different physical parameters. The user can choose machines from a list of available hosts, apply a preprocessor on indicated input parameters or explicitly set input parameters for each process, and start the jobs on the respective machines.

For true parallel applications, mxpl allows the user to select the respective nodes and creates the host list or procgroup file for various parallel programming packages.
With large workstation clusters where many jobs run simultaneously, it may be crucial to get an overview of the loads of the available machines. A visual load distribution tool is supplied to help the user in finding machines on a network that are available for computation.

5. Visualization

The exploration of materials requires a detailed analysis of the simulation data. Of course, the simulated configurations present an overwhelming amount of information. Therefore mxpl provides for advanced visualization and user--interaction methods beyond scalar quantities and simple two--dimensional plots. The tools run on all computer systems offering the X-Window system, including X-terminals and personal computers with X-Server software. Expensive high--end graphics workstations are not neccessary, but their special hardware can be used, if neccesary. Also, regarding the quality of the graphics, the user is not limited by his hardware. One has the choice between several display qualities, starting from fast line graphics up to slower photo--realistic rendering. As a typical application, some aspects of the molecular force microscope MFM will be discussed now.

The fastest visualization method uses coloured lines for the bonds and circles for the atoms of a three--dimensional configuration and orthogonal or perspective projection. The user can rotate, move and zoom the whole configuration. Using the mouse,

one can select an atom or a chain to get the coordinates of the respective atom. Additionally, one can switch between visualizations with or without periodic boundary conditions, which is very useful for the exploration cof polymer chain properties. The second method of visualization uses a surface representation with hidden surface detection and an illumination model. Figure 3 shows bonds represented as cylinders and atoms as spheres. In general, without special graphics hardware this method is quite slow. Therefore we developed fast software rendering techniques for the graphics library of mxpl and a suitable refresh rate can be obtained on todays workstations and personal computers even in this display mode without a hardware accelerator. In general, these two methods are used during on--line exploration of materials. For printing or production of slides, our built-in raytracer [6] produces photorealistic images (see figure 3).

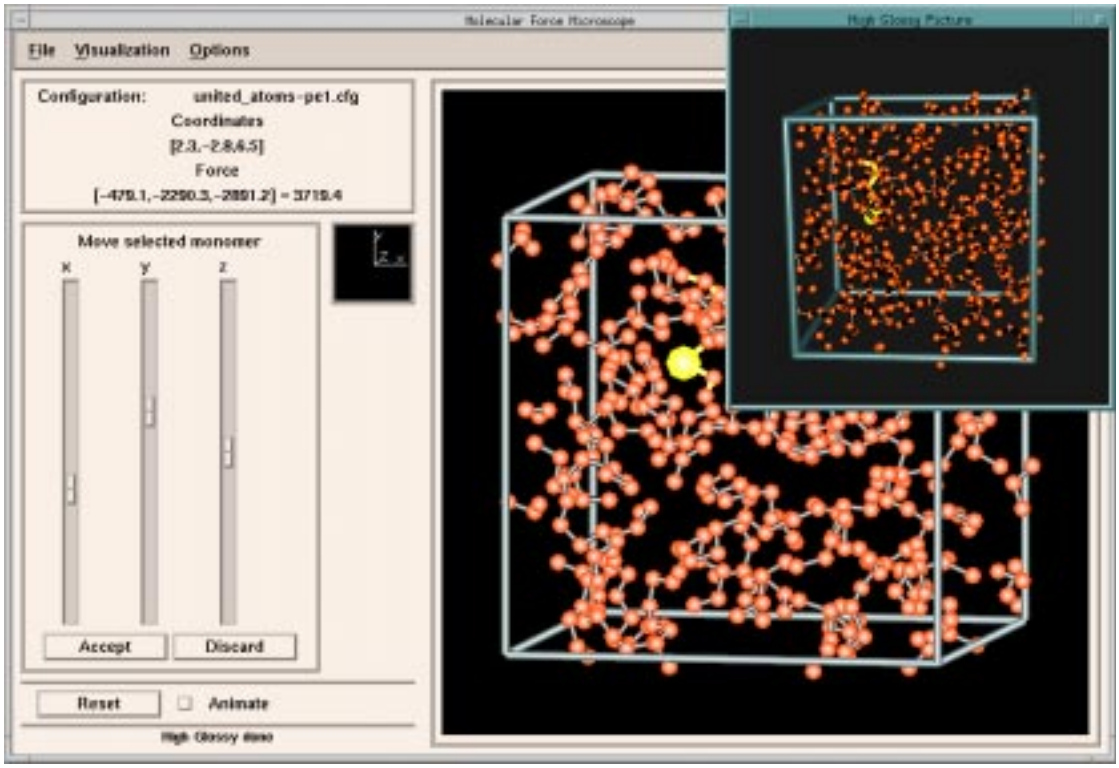

Figure 3: Different display qualities used in the molecular force microscope MFM of the Materials Explorer

Of course, mxpl and MFM makes feasible an advanced kind of user interaction: The on--line modification and visualization of configurations. Figure 3 shows a snapshot of such a modification. An atom was selected by the user and then shifted. The new positions of the atom and the bonds are visualized on the right, whereas the new particle forces are shown numerically on the left. It is possible, to extend this kind of data representation. For example, the forces can be shown as arrows within the configuration or velocities may be coded with the colors of the atoms.
At this stage of material exploration, the new understanding may result in new concepts of visualization, which will be implemented into mxpl also.

6. Discussion

The use of a graphical user interface for starting simulations simplifies routine procedures (like e.g. in a commercial environment where the same simulation will be executed with the same parameters and different input data over and over again). On the other hand, the high flexibility and adaptibility of the user interface makes it invaluable in a research environment with a lot of ever changing applications. The Materials Explorer is a step in the direction of an open development package where industry, software house and university cooperated to tailer the application to the daily needs in a research and development environment.

References

[1] For recent reviews see: K. Binder, in **Phase Transformations in Materials**, ed. P. Haasen, Materials Science and Technology, Vol. 5, VCH Verlag, Weinheim, 1991
and S. Komura and H. Furukawa, eds. **Dynamics of Ordering Processes in Condensed Matter**, Plenum Press, New York, 1991

[2] D.W. Heermann, **Computer Simulation Methods in Theoretical Physics,** 2nd ed., Springer Verlag, Heidelberg, 1990

[3] K. Binder and D.W. Heermann, **Monte Carlo Simulation in Statistical Physics: An Introduction**, Springer Verlag, Heidelberg, 1988

[4] D.W. Heermann and A.N. Burkitt, **Parallel Algorithms in Computational Science**