Computer Physics Communications 124 (2000) 60–75

Computer Physics Communications

www.elsevier.nl/locate/cpc

An interactive simulation program for visualizing complex phenomena in solids

J. Merimaa, L.F. Perondi *,[1], K. Kaski

Laboratory of Computational Engineering, Helsinki University of Technology, P.O. Box 9400, FIN-02015 HUT, Finland

Received 16 February 1999; received in revised form 2 June 1999

Abstract

In the present article an interactive simulation program illustrating grain boundary and fracture phenomena in solids is described. The dynamical behaviour of a two-dimensional Lennard-Jones model system under stress, with either a grain boundary or an initial crack, is simulated through a molecular dynamics algorithm. All parameters defining the system and the dynamical load are set through a graphical user interface. A run-time representation of the system is displayed on a graphics window, which has been endowed with magnification and other visualization aid tools. The program runs on a UNIX-X11 Window System platform. The graphical part relies on the MOTIF library. The program has been devised for illustrative purposes. It displays the main elements of an interactive simulation and may be regarded as giving an illustration of the concept of interactivity. Due to the power of run-time animations in conveying ideas and concepts, it may prove to be useful, as well, as an instructional tool.

PACS: 07.05.T; 79.20.A; 01.50.H; 62.20M
Keywords: Computer modeling and simulation; Grain boundaries; Fracture; Dislocations; Molecular dynamics; Graphical interface

PROGRAM SUMMARY

Title of program: Fracture

Catalogue identifier: ADKS

Program Summary URL:
http://www.cpc.cs.qub.ac.uk/cpc/summaries/ADKS

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland

Computer for which the program is designed and others on which it is operable: *Computers:* DEC ALPHA 300 and equivalent or superior machines

Installations: Lab. of Computational Engineering, Helsinki University of Technology, Helsinki, Finland

Operating systems under which the program has been tested: UNIX

Programming language used: C with MOTIF and standard libraries

Memory required to execute with typical data: Minimum of 2 Mbytes but depends on system size

No. of bytes in distributed program, including test data, etc.: 37 629 bytes

Distribution format: uuencoded compressed tar file

* Corresponding author; e-mail: perondi@lce.hut.fi.
[1] Permanent address: Instituto Nacional de Pesquisas Espaciais – INPE, P.O. Box 515, 12-227-010 São José dos Campos SP, Brazil.

0010-4655/00/$ – see front matter
PII: S0010-4655(99)00264-7

Keywords: Lennard-Jones, molecular dynamics, fracture, crack, dislocations

Nature of physical problem
In a typical research activity involving simulations of complex physical systems, one is often faced with a 'research loop', which may be loosely described as follows: definition of input data; running of a simulation, outputting data to a storage disk; post-processing of the accumulated data for extracting relevant information with the eventual aid of graphics and numerical packages; definition of new input parameters and, closing the loop, starting a new simulation run. Advances in computer hardware, mainly faster processors, and the increased availability of software development tools, with plenty of graphics resources, have been markedly increasing the feasibility of alternative solutions for improving the above scheme.

Method of solution
Interactive simulations offer an attractive option for the study of systems in which a great deal of feedback, in the sense described above, is required. They are also ideal for teaching purposes. The present program gives an illustration of an interactive simulation environment. The dynamical behaviour of a two-dimensional Lennard-Jones 'solid' under stress, with either a grain boundary or an initial crack, is simulated through a molecular dynamics algorithm.

Restrictions on the complexity of the problem
The interactiveness concept strongly relies on the display of an animated graphical representation of the system. The quality of the animation depends on the rate with which frames are displayed on the graphical display window. This rate, in its turn, depends on the iteration time of the simulation algorithm. Hence, the size of the systems which may be investigated, while maintaining an acceptable animation quality, depends on the machine processor. Systems with up to 50 000 atoms give fair animations for the target machine given above. Subjecting the system to strain rates sufficiently high for inducing atomic displacements of the order of the mean inter-atomic distance in one integration step will cause the program to collapse. Studies at such high rates will require modifications of the MD algorithm. The implementation of an adaptive integration method will, most possibly, be enough for most applications.

Typical running time
The typical running time is machine and system size dependent.

Unusual features of the program
Although the program has been originally designed for illustrative purposes, its final version incorporates features which, we believe, render it in a good standard for research work in two-dimensional models, provided a better treatment is given to thermal effects. In particular, it has been endowed with a control button which allows the user to generate an instantaneous 'snapshot' of the graphics screen. The image is stored in a bitmap file (X11 standard), created under the same directory in which the program is running.

LONG WRITE-UP

1. Introduction

The application of computers and computational techniques to the study of physical phenomena is by now a well-established and important branch of modern science. The course of its development has closely followed the improvement on the availability of hardware and software resources. In the wake of these developments, applications evolved, in materials research, for example, from the numerical solution of small scale systems of algebraic equations to the current *multi-million atoms* molecular dynamics (MD) simulations of various phenomena in liquids and solids [1–3]. The current availability of computer resources, with desktop machines exhibiting the performance of the supercomputers of yesterday, has been stimulating new developments at an impressive rate [4]. Confining ourselves to examples directly related to the subject of this article, we quote the development of interactive simulations of many-particle systems [5] and of specially-tailored techniques and tools for the analysis of data arising from the simulation of complex physical phenomena [6,7].

Conceptually, interactive simulations may be envisaged as the association of visualization tools with a numerical simulation code in a message-oriented or event-driven environment. They present very distinctive advantages when compared to *conventional* simulations. In the latter, the way simulation data are generated and analyzed may schematically be described as follows: define initial parameters, run the simulation and output the data to a storage medium and in a post-processing phase analyze the results with the eventual aid of numerical and graphics packages; eventually, define a new set of initial parameters and perform a new run. In an interactive simulation environment, otherwise, the simulation data may be output directly to a graphics environment and simulation parameters may be changed at any time. In a many-particle system, for instance, it is then possible to visually accompany the time evolution of the system and to impose changes to it at run-time by assigning new values to important parameters. This ability to drive the system into new states, guided by a visual feedback, creates a rich research environment, which offers several pos-

sibilities, ranging from a sensitivity analysis of relevant parameters to the emulation of situations that may give valuable insight into phenomena of great complexity.

Simulations aimed at the study of the mechanical properties of solids, for the richness of the phenomena they exhibit, are natural candidates for being endowed with an interactive interface. In the present article we illustrate some of the potentialities of an interactive simulation environment by considering phenomena associated to fracture and grain boundary in solids. More specifically, we will describe a program specially designed for illustrating such phenomena and give a concise guide about its use. Despite the simplicity of the implemented model – a two-dimensional Lennard-Jones solid – it has proved to be rich enough to illustrate important phenomena such as emission and propagation of dislocations, ductile and brittle fracture behaviour and grain boundary relaxation.

The program has been developed as part of an effort to assess the relevance of the interactive approach to molecular dynamics simulations of solid state phenomena. Despite this background, the presentation given here is centred on a description of the developed program and its use, rather than on a description and analysis of the interactive approach itself. Only those aspects of interactivity which are relevant to an understanding of the program have been considered in some detail. The program has been devised for illustrative purposes. It displays the main elements present in an interactive simulation and is intended to give an illustration of the value of interactivity in computer simulations. Owing to the physical model implemented and the help animation may give in the explanation of physical phenomena, it may prove to be useful, as well, as an instructional tool in materials science and solid-state physics courses.

The article is organized as follows. In Section 2 we give the main definitions and some physical motivations related to the study of grain boundaries and fracture in solids. Main technical aspects relative to the implemented algorithm are also addressed. Section 3 is dedicated to a general description of the program, its structure and the graphical interface. Some examples are described in Section 4. Final comments and conclusions are given in Section 5.

2. Model

In this section we address some conceptual aspects relative to the program. The exposition is divided in two parts. Firstly, we give a brief introduction to physical concepts related to grain boundaries and fracture in solids. Only very general aspects, which are related to the implemented model and the examples in Section 4, are contemplated. Detailed information may be found in the references. Since we will be dealing with a two-dimensional model we emphasize the role of dimensionality. Secondly, an overview of the simulation model and a description of the main visualization tools are given in Sections 2.2 and 2.3, respectively.

2.1. Fracture and grain boundaries

The subjects of fracture and interfaces in solids, together with their associated phenomena, are currently among the central research themes in materials science.

Single-phase interfaces, such as those present in polycrystalline materials, are generally referred to as grain boundaries [8]. Understanding the structure and the dynamics of grain boundaries and their interaction with defects is one of the major challenges in setting up realistic models for the mechanical behaviour of materials. While in three dimensions there are five degrees of freedom in the formation of a boundary between two crystallites, in two dimensions (2D) there are only two. In this latter case, for equal crystallites, it suffices to define the rotation of each crystallite with respect to the interface line to characterize any possible boundary. Such boundaries are termed tilt boundaries. In three dimensions (3D), a tilt boundary, paralleling the 2D case, may be thought as formed by the rotation of two initially equally oriented crystallites through a common rotation axis. Tilt boundaries may be classified into symmetric or asymmetric, depending upon whether each crystallite is or is not the mirror image of the other.

The imperfections at a low angle 3D tilt grain boundary may be interpreted as *edge dislocations* [9]. A perfect edge dislocation is a line defect associated with the abrupt end of a plane of atoms inside the bulk region of an otherwise perfect crystal. The dislocation line is given by the faulty edge of this plane. Simi-

larly, imperfections at a 2D grain boundary may be regarded as the 2D equivalent of an edge dislocation. In complete analogy with the 3D case, 2D edge dislocations are formed by the abrupt end of a line of atoms in the internal part of a 2D layer of atoms. The 2D edge dislocation is the defect associated with the tip of the faulty line. Appendix 5 gives further details about 2D dislocations in a triangular lattice. Great interest exists in the microscopic structure of grain boundaries. It depends on geometric parameters, such as the orientation of the crystallites, as well as on the interatomic potential. Studies of grain boundary structure are generally carried out by initially generating the boundary in accordance with the geometrical constraints and then letting the resulting system to relax.

The understanding of fracture phenomena is of great relevance from both the scientific and technological points of view [10]. The literature on the subject is quite extensive [11]. We will limit ourselves to a brief description of the main facts about fracture in crystalline materials. The way a macroscopic fracture springs up and evolves in a strained sample is strongly dependent on the *brittleness* of the solid. In very brittle materials a crack with a sharp tip propagates at very high speeds, leading invariably to cleavage of the sample. The crack path may show irregularities, the origin of which has been the subject of some debate [12,13]. At the other extreme, in ductile materials, a crack may not propagate at all. Up to comparatively high strain levels, there is only the emission of dislocations, the net effect of which is to cause *blunting* of the crack tip. Most of the current research focus on the modeling of the ductile/brittle behaviour of the crack tip in terms of a microscopic, atomistic level, model potential. Two-dimensional models have attracted a great deal of attention [2,3,14–16].

The program, object of this article, allows the definition of a two-dimensional system containing either a grain boundary or an initial crack (or both). Both symmetric and asymmetric tilt boundaries may be generated. The implemented resources permit the simulation of grain boundary relaxation and the study of the effect of stress and temperature on it. The fracture-related implementation permits the placing of an initial crack, with chosen rectangular dimensions, anywhere in the simulation area. The main objective is to follow what happens to the resulting structure when it is subjected to strain at a given temperature. To this effect, the user may specify at run-time a given strain level and the rate at which it is applied. Also, making use of run-time controls, the user may set, at any moment, a new temperature value. The program also features controls which allow the study of the effect of changes in the inter-atomic potential on the the grain boundary and fracture phenomena. Next we give details of the simulation and visualization algorithms.

2.2. Simulation model

Inter-atomic interactions have been modelled by a Lennard-Jones pair potential

$$U(r) = e\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right), \tag{1}$$

where r is the inter-atomic distance, σ defines the equilibrium distance r_e through the expression $r_e = 2^{1/6}\sigma$ and e fixes the value of the potential at the equilibrium distance. Owing to the symmetry of the potential the equilibrium atomic arrangement is a triangular lattice. A conventional molecular dynamics (MD) algorithm has been implemented [17,18]. In order to speed up the computations the interaction range has been limited in the usual way, by setting the potential to zero beyond a maximum inter-atomic distance $r = r_c$. To avoid an anomalous behaviour of the force at $r = r_c$, a linear term with the form $(r - r_c)\frac{dU}{dr}|_{r=r_c}$ has been subtracted from U. For reasonable values of r_c this approximation has the minor side effect of slightly changing the equilibrium distance. The user may adjust the values of e and r_c in order to endow the model with a brittle or ductile character. Generally, large values of e and small values of r_c favour a brittle behaviour.

The simulations are carried out in the micro-canonical ensemble. Temperature is equated to kinetic energy, even when the system is far from thermodynamical equilibrium. In each run the system starts with a given instantaneous value of the kinetic energy, which is fed in by assigning an initial velocity to each particle. The velocity vectors have equal magnitude and are given a random orientation. The excess kinetic energy in any of the directions is then removed. The user may change at any moment the energy content of the system by specifying a new instantaneous value for the kinetic energy. Arrow buttons are provided for

this purpose. The new value is enforced by simultaneously re-scaling the magnitude of the velocity vectors of all particles. Alternatively, the temperature during a whole simulation run may be kept below a certain value (the default value is 20 K) by checking a box control specially set for this purpose. The operative mechanism triggered by this control is equivalent to a continual pressing of the down temperature arrow button when the temperature exceeds the threshold value in effect. These features have been implemented with the purpose of controlling the total energy in the system, which can substantially increase when the system is strained. We emphasize that this scheme gives a poor representation of thermal effects. A proper treatment would involve the use of some intrinsic mechanism for emulating the heat exchange with an external heat reservoir [18].

When attempting to study the dynamics of bulk phenomena through simulations in a finite system, eventual boundary effects have to be taken into account. In the present simulations, reflection of strain waves at the boundaries are the main visible effect. Reflected waves may have a noticeable influence on the way cracks evolve, mainly due to interference. They may also significantly affect the way relaxation takes place at grain boundaries. No special features have been incorporated in order to minimize such effects. Additionally, since no intrinsic dissipative mechanism has been implemented wave coherency is longer lived, giving enhanced interference effects. The only available way for limiting the influence of such effects is to control the total energy content of the system in the manner discussed above.

Strain is applied to the system by changing its x-dimension (parallel to the screen bottom edge) by an amount and at a rate specified by the user. Practically, this is accomplished by rigidly moving, at a constant velocity, the atoms belonging to a few columns of the external vertical (y-direction) borders of the simulation area. In order to avoid undesired *edge reconstructions*, the atoms in these 'fixed' borders are not allowed to move in the x-direction as a result of interactions. Their movement in the y-direction, however, is completely unconstrained. The user may choose the width of the fixed borders. The *fixed border* feature may be switched on/off at any moment. Free boundaries are important, for instance, when studying relaxation at grain boundaries.

2.3. Feature highlighting

Extracting the desired information from a batch of data is an important aspect when setting up an interactive simulation. Interesting phenomena are usually *hidden* in the data and a feature extraction algorithm has the objective of searching for and isolating their data structure for subsequent processing.

In the present case, the most direct way of displaying system features is to show a graphical representation of the atoms on the screen. This is, in fact, the main graphical output of the code at hand. Visualization of phenomena may be enhanced by secondary tools, which operate directly on the graphics screen. Independent transverse and longitudinal (x- and y-directions, respectively) magnification (*zooming*) tools are examples. Their implementation is quite straightforward and they proved to be ideal for visualizing line defects. By a careful tuning of the magnification factor they may also greatly improve the visualization of extended phenomena, such as strain waves. Further insight into the complexities of the exhibited phenomena may be achieved by arranging the particles into groups according to some criterion and then differently representing each group on the screen. A common scheme is to assign different colours to different groups. The code includes one such feature. The atoms are stratified into categories according to the value of their kinetic energy. A relative scale, in which the average kinetic energy is set to the midpoint, is used for this purpose. Particles are then displayed on the screen according to a colour scheme which associates light red to the most energetic particles and black to the least energetic ones. The intermediate cases are shown with different shades of red. Such scheme enhances the visibility of phenomena taking place near boundaries and defects.

In addition to the particle representation discussed above, a *field representation* has also been implemented. The user may switch between the two representations at any moment. In the latter representation the velocity field at different length scales is displayed. The length scale may be continuously changed from the *microscopic* to the *macroscopic* (i.e., comparable to the size of the system) limits. In the *microscopic limit*, an arrow in the direction of the velocity of a particle is drawn directly at the particle position. This gives a clear view of the dynamical behaviour of the

microscopic velocity field. The representation at larger scales is constructed through an averaging process: the simulation area is covered with a grid having a lattice parameter size specified by the user; an average position and velocity is then computed for each grid square by averaging over the particles which are within its borders; finally, at each of the computed average positions an arrow giving the direction of the average velocity is drawn on the screen. The final picture is a larger scale representation of the velocity field.

3. Description of the interface

As already mentioned, the program consists essentially of three modules: an MD simulation algorithm, a graphics window for output of data and a user interface for input of parameters. In a very schematic description, we may say that these modules have been integrated in an event-driven environment which handles the exchange of information between them, and processes any actions requested by the user at runtime. Fig. 1 schematically illustrates the flow of information.

The program has been written in C and developed for an X11 Window System platform. All graphics are based on the MOTIF library. A great deal of UNIX workstations make use of this platform. A detailed description of the programming environment may be found in Refs. [19–21]. In this section we limit the discussion to a description of the user interface.

Figs. 2 and 3 show the two initial windows which show up at startup. The window shown in Fig. 2, labelled 'Parameters', allows the user to select the 'fixed' parameters of the simulation. These are general parameters which define the main characteristics of the simulated system, and which are held constant throughout a simulation run. They are organized into groups, under a descriptive header. The groups, in their turn, are aligned into columns. Table 1 gives a general description of each group.

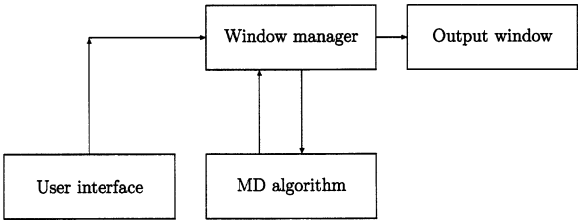

Fig. 1. Schematic structure of an interactive simulation program. The window manager handles the exchange of data among the different modules.

The second window, shown in Fig. 3 and named 'Fracture', contains, in a broad sense, four categories of elements: a text window displaying run-time data; a graphics window giving a run-time graphical representation of the system; control widgets giving the user some control on the way data are displayed on the graphics window and control widgets allowing the change of simulation parameters at run-time. All elements in this window are described in Table 2.

The program has, essentially, two operating modes: a initialization mode and a running mode. In the initialization mode, the user defines the general characteristics of the system to be simulated by setting the controls in the window 'Parameters'. Notice that the graphics window, in the window 'Fracture', is active even in this mode and displays an updated representation of the system. The text pane is also active. When the user presses the ***OK*** button at the lower right corner, only the window 'Fracture' remains open. Pressing the ***Run*** button in this latter window causes the program to enter the running mode. In this mode the simulation algorithm is active and the graphics window is updated at a rate given by the ***Update step*** control widget setting.

The user may suspend/resume the simulation at any moment by pressing the ***Pause/Run*** button. When the button ***New*** is pressed, the program is suspended and the window 'Parameters' is popped up. At this point, the user may either select new settings and begin a new simulation run by pressing the ***OK*** button or cancel the operation, through the ***Cancel*** button. The pressing of either button *pops down* the window 'Parameters'. When the button ***Quit*** is pressed all windows are closed and the program ends. Finally, the button ***Snapshot*** allows the user to dump to a file the current image in the graphics window. The image is stored in a file with X11 standard bitmap format. Consecutive files generated during the same session are labelled with consecutive letters. The files have names with the format *Fracture_L.img*, where *L* stands for any letter of the alphabet, either capital or lower case.

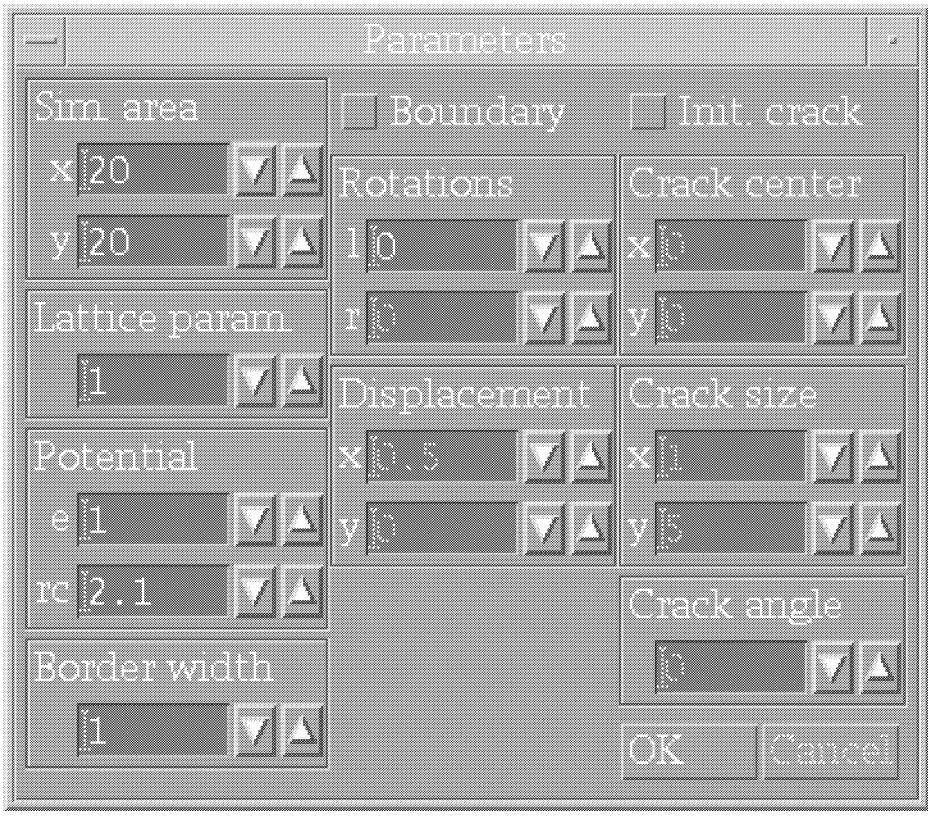

Fig. 2. Window 'Parameters'. Initialization window with controls for definition of the 'fixed' parameters of the simulated system.

Normalization constants have been chosen such that the default (initial) settings for the lattice parameter ($a = 1$) and the potential ($e = 1$) reproduce a LJ pair-potential appropriate for the study of copper, as given in Ref. [22]. Distances are given in units of the lattice parameter for copper ($a_0 = 3.60 \times 10^{-10}$ m [23]); time has been normalized to one picosecond ($t_0 = 10^{-12}$ s); the integration step is equal to four femtoseconds, i.e., each MD step corresponds to the elapsing of one such time interval; the elapsed time displayed in the main window is given in units of one femtosecond; energies are given in units of $m_0(a_0/t_0)^2 = 1.37736 \times 10^{-20}$ J, where m_0 is the mass of one atom of copper ($m_0 = 1.055206 \times 10^{-25}$ kg [23]). Finally, temperatures are given directly in Kelvin units. Table 3 gives a summary of the main equations and units.

Details about the program files and command line options are given in Appendices 5 and 6.

4. Examples

In this section we describe and comment on some few illustrative examples. The presentation has two purposes, namely giving a brief introduction to the use of the program and displaying some of the potentialities of an interactive simulation environment.

4.1. Tilt grain boundary

By adjusting the settings according to the values given in Table 4 under the heading *symmetric* and letting the resulting system to relax, a 12° symmetric tilt boundary is formed. Please, note that the ***Fix edges*** check box in the Strain pane should be unchecked in this example. The stable configuration shown in Fig. 4 is reached after 21.2 picoseconds (21 200 in the time display). Note that the defects appearing in the boundary region are 2D dislocations, of the same type

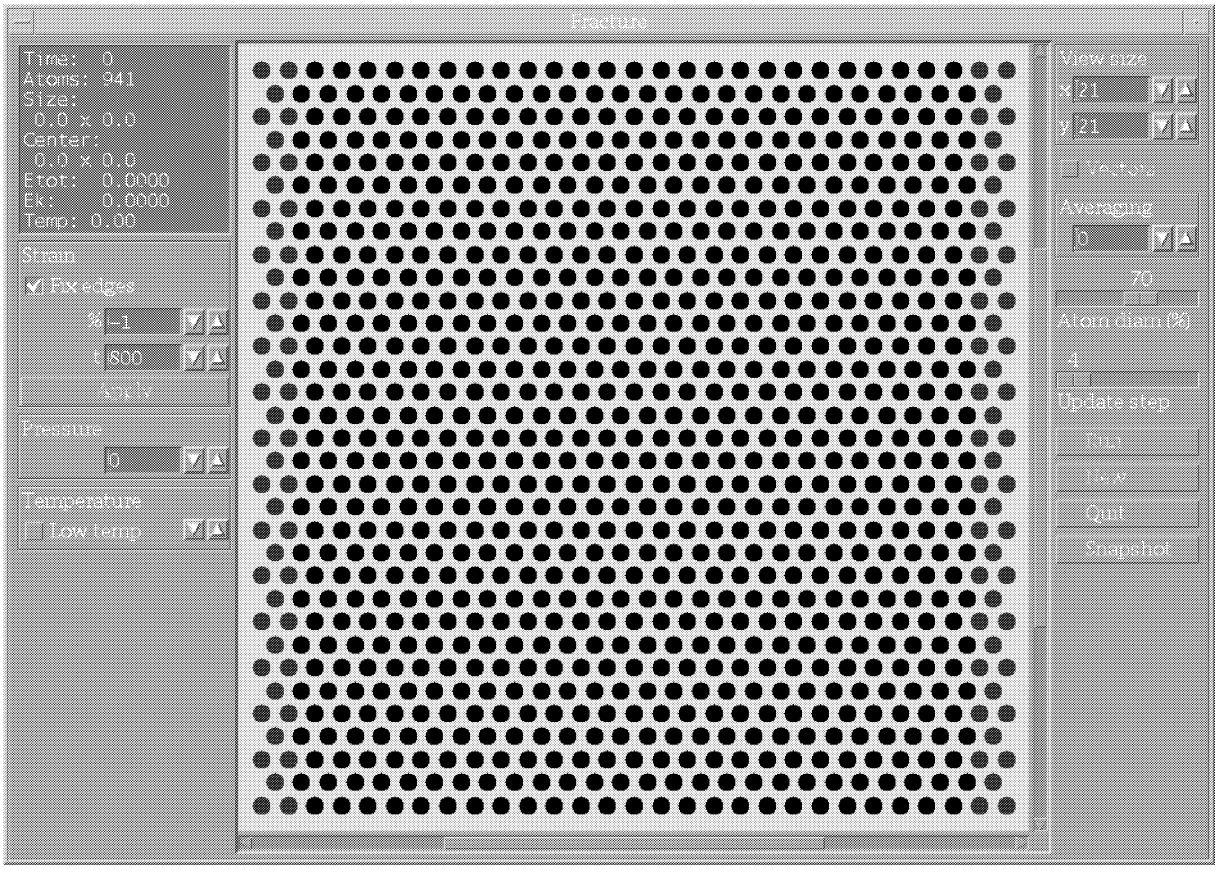

Fig. 3. Window 'Fracture'. Contains the graphics window, the text pane and the controls which allow the change of parameters at run-time.

as the ones described in Appendix 5. The observed vertical piling of the dislocations, highlighting the position of the boundary, may be interpreted as a result of the interaction between similar dislocations [9]. A close look at one of the dislocations may be achieved by setting $x = 5$ and $y = 5$ in the ***View size*** pane and setting the ***Atom diam.*** scale to 93. If the visible dislocation is then repositioned at the centre of the graphics window, through the sliding bar controls, the obtained image will resemble the one shown in Fig. 5.

The settings under the heading *asymmetric* in Table 4 give an initial configuration for the formation of a 9° asymmetric tilt boundary. After about 28.0 picoseconds, the structure relaxes to the configuration shown in Fig. 6. Most of the dislocations are now on top of a line which deviates from the vertical by an angle around 4.5°. If we take the aforementioned line as reference it is easily seen that both crystallites are symmetrically oriented with respect to it. The final configuration, therefore, is again that of a symmetrical tilt boundary. This is an example of grain boundary movement. The vacancies seen at the bottom part are due to the non-conservative movement of some of the dislocations (see Appendix 5). A close look at the vacancies and dislocations may be achieved with the *zooming* settings given in the previous example.

Grain boundaries tend to become unstable with increasing temperature. One may get some insight into this phenomenon by repeating the above examples and imposing changes to the temperature of the system after the initial equilibrium boundary has been formed. Also, the application of stress, be it compressive or tensile, causes changes in the equilibrium configura-

Table 1
Description of the controls in the window 'Parameters'. The symbols in parenthesis give the unit in which the corresponding control is expressed. Please refer to Table 3 for a definition of the used units

Column 1 – General parameters	
Sim. area	
x, y	initial x- and y-dimensions of simulation area (a_0).
Lattice parameter	lattice parameter size.
Potential	
e	minimum value of the LJ pair-potential (Eq. (1)) (u_0);
r_c	cut-off radius of the pair-potential (a_0).
Border width	defines a stripe at the external longitudinal borders with the specified width, inside which all atoms are constrained to move only along the y-direction; when strain is applied, all atoms inside this stripe are rigidly moved with the specified strain rate (a_0).
Column 2 – Definition of the boundary	
Boundary	when the box is checked the system is split into two independent sub-lattices, right and left, of equal area. Since each sub-lattice may be rotated independently of the other, boundaries with an arbitrary orientation may be formed.
Rotations	
l, r	angle of rotation in degrees of the left and right sub-lattices, respectively.
Displacement	
x	boundary gap as measured from the centre of the closest edge atoms (a_0);
y	*microscopic* y-displacement of one sub-lattice with respect to the other (a_0).
Column 3 – Crack geometry and positioning	
Init. crack	checking of the box causes a rectangular crack to be inserted into the system. This is the initial crack for a fracture 'experiment'. The remaining controls in this column define the crack position in the simulation area, its rectangular dimensions and the angle its x-edge makes with the x-edge of the simulation area. All distances are in units of (a_0).
Crack centre	
x, y	x- and y-positions of the crack centre (a_0).
Crack size	
x, y	x- and y-lengths of the crack area (a_0).
Crack angle	angle, in degrees, between the x-edges of crack and simulation area.

tion of the boundary. These changes may be followed by checking the ***Fix edges*** box and setting values for the total strain and the time duration of its application in the corresponding controls. All these controls are in the Strain pane.

4.2. Brittle versus ductile fracture

Here we briefly consider two examples which illustrate the limiting cases of brittle and ductile fracture. Table 4, under the heading *fracture*, gives the settings for both cases. In both cases, strain is applied at the very beginning of the simulation.

In the ductile fracture example, it is observed that the crack tip initially advances for a few atomic layers and then a pair of dislocations is emitted. After this event, the crack tip position remains practically stationary and undergoes an accentuated blunting as further dislocations are emitted. Several processes are triggered by the dislocation emission. Among them, the heating up of the system and the generation of strain waves are particularly noticeable. These effects may be better appreciated by repeating the *experiment* but now accompanying the unfolding of events in the *velocity field* representation (***Vectors*** button). Figs. 7 and 8 display the configuration after 6.27 picoseconds

Table 2
Main Elements of the window 'Fracture'. The symbols in parenthesis give the unit in which the corresponding control is expressed. Please refer to Table 3 for a definition of the used units

Text window and its elements	
Text window	displays static and run-time data about the system. The displayed data are:
Step	elapsed time ($10^{-3} \times t_0$);
Atoms	total number of atoms in the system;
Size	x- and y-dimensions of the simulation area (a_0);
Centre	coordinates of the position in the simulation area which is at the centre of the graphics window (a_0);
Etot, Ek	total energy and kinetic energy per particle, respectively (u_0);
Temp	temperature in Kelvin degrees.
Graphics window and related elements	
Graphics window	displays a run-time graphical representation of the system. The sliding bars allow repositioning the simulation area in the graphics window.
View size pane	the two controls in this pane define the scale factors for the x- and y-axis. Their precise meaning are: x- and y-dimensions of the part of the simulation area which is being shown in the graphics window (a_0).
Atom diam. control	allows the user to change the size of the objects (vectors or circles) displayed in the graphics window;
Vectors box	when checked the graphics window display a *field representation* (see text) of the system;
Averaging control	coarse grid size for the vector representation (Section 3) (a_0).
Run-time parameters	
Strain pane	the controls in this pane define the total strain and the strain rate to which the system will be subjected when the **Apply** button is pressed.
Fix edges	when checked the external longitudinal borders are fixed (as described in Section 3);
%	total strain;
t	time interval in which the strain defined in the control above is applied to the system (in femtoseconds) ($10^{-3} \times t_0$).
Pressure	defines the intensity of a force applied to each particle at the transversal borders (f_0).
Temperature	allows the user to control the kinetic energy content of the system (as discussed in Section 3).

Table 3
Normalization constants and main equations. The variables cut, a and e hold the settings of the corresponding controls in the window 'Parameters'

Main equations:	$r_c = cut \quad a/\sqrt{2}$
	$\sigma = \left(\frac{1}{2}\left(\frac{1-(1/cut)^7}{1-(1/cut)^{13}}\right)\right)^{1/6} a/\sqrt{2}$
	$\frac{d^2\boldsymbol{r}}{dt^2} = 110.09e\left(\left(\frac{\sigma}{r}\right)^{13} - \frac{1}{2}\left(\frac{\sigma}{r}\right)^{7} - \left(\left(\frac{\sigma}{r_c}\right)^{13} - \frac{1}{2}\left(\frac{\sigma}{r_c}\right)^{7}\right)\right)$
	$U(r) = 9.17e\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} - 12\frac{r-r_c}{\sigma}\left(\left(\frac{\sigma}{r_c}\right)^{13} - \frac{1}{2}\left(\frac{\sigma}{r_c}\right)^{7}\right)\right)$
Units:	$a_0 = 3.608 \times 10^{-10}$ m (unit of distance)
	$t_0 = 10^{-12}$ s (unit of time)
	$f_0 = 3.806 \times 10^{-11}$ N (unit of force)
	$m_0 = 1.055 \times 10^{-25}$ kg (unit of mass)
	$u_0 = 1.37736 \times 10^{-20}$ J (unit of energy)

Table 4
Settings for the examples discussed in the text

		Grain boundary	
		Symmetric	Asymmetric
Sim. area	x:	80	80
	y:	40	45
Lat. param.		1.0	1.0
Potential	e:	1	1
	rc:	2.1	2.1
Rotations	l:	6	9
	r:	−6	0
Displacements	x:	0.5	0.6
	y:	0.6	−0.3

Obs.:
In the grain boundary examples the check box **Fix edges** should be unchecked. Convenient settings for the **View size** pane for these particular examples are $x = 32$ and $y = 43$.

		Fracture	
		Ductile	Brittle
Sim. area	x:	60	60
	y:	20	30
Lat. param.		1.0	1.0
Potential	e:	1	20
	rc:	2.1	2.1
Crack center	x:	0	0
	y:	7.5	12.5
Crack size	x:	1.0	1.0
	y:	5.0	10.0
Crack angle		0	0

Obs.:
In the first example, appropriate settings for the strain rate are: % 5, t 8000. Corresponding settings for the second case are % 10, t 16000. In both cases, for a close look at the crack tip, convenient settings of the **View size** pane are $x = 30$ and $y = 30$, with repositioning of the crack at the centre of the graphics area.

in the particle and field representations, respectively. A close view of the events taking place near the crack tip may be achieved through the magnification tools described in Section 3 (***View size*** pane). For instance, in the present example, the settings *30* and *10* for the x- and y-directions, respectively, enhances the visualization of the family of *most dense lines* parallel to the x-direction. With these settings and positioning the crack tip at the centre of the display window it is possible to follow in great detail the events preceding the emission of the pair of dislocations. The enhanced visibility of dislocations and their movement is one of the special features of this *zooming* scheme. Toggling between the particle and vector representations further enhances the visibility of the phenomena taking place.

The brittle fracture *experiment* is conducted in a similar way as in the ductile case, except for one important difference – in order to enhance the brittle behaviour the extra heat generated by the *breaking* of bonds must be continually removed. This is accomplished by checking the box ***Low temperature***. Contrary to the ductile case, the crack tip now continues advancing until the test sample undergoes a complete breakage. Figs. 9 and 10 show the system configuration after 7.68 picoseconds in the particle and field representations, respectively. Here, as in the previous case, a rich variety of phenomena is observed as the system is strained. As examples, we quote two: as the crack tip advances dislocations are emitted along the tip's propagation direction, and each time a dislocation is emitted, the direction in which the crack propagates switches from one *easy slip* direction to another. It is instructive repeating this same experiment for a smaller value of the cut-off radius of the potential (for example, $r_c = 1.1$). It will be observed that no dislocations are emitted in this case. The final breakage pattern resembles that of a shattered fragile flat sample.

5. Final comments and conclusions

In the above examples, some of the advantages of the interactive approach are already apparent. For instance, in the symmetric tilt boundary example, if data

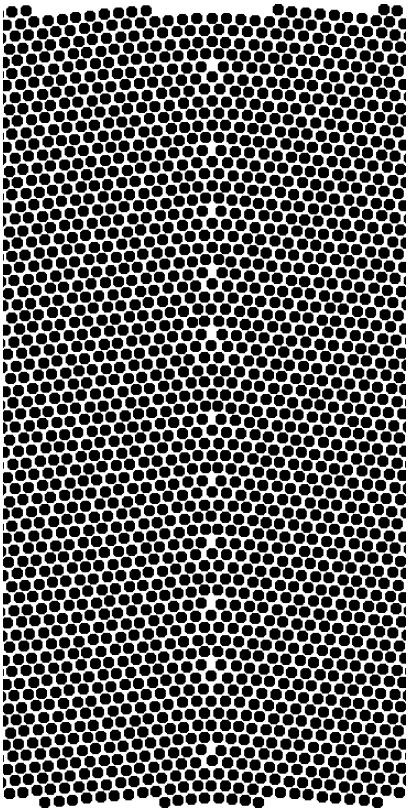

Fig. 4. Symmetric tilt boundary. The configuration shown corresponds to the example discussed in the text.

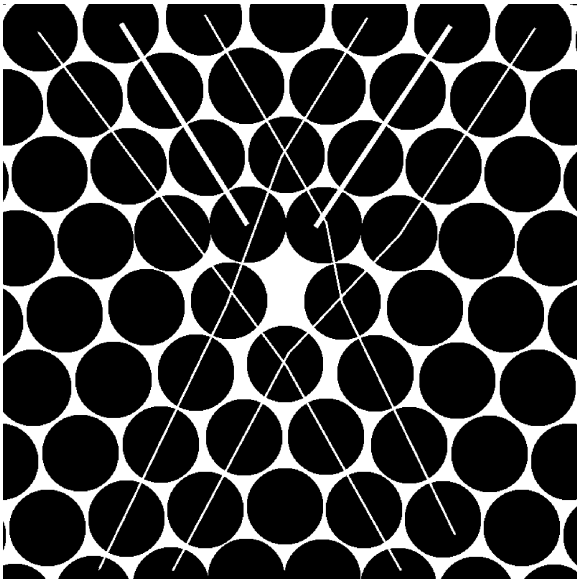

Fig. 5. Enlarged view of one of the dislocations found in the symmetric tilt boundary example. Thin lines are a guide for the eye in singling out the extra *rows*. These are marked with thick lines.

were dumped to a disk for a later full animation, the storage of the coordinates of all particles for all time steps would require something around 500 Mbytes of disk space. This value would double if velocities were also stored. In the interactive approach, otherwise, the direct flow of data to the screen combined with the possibility of storing configurations in a selective way keep the required disk space at a minimum. Another feature of this specific implementation is the easy with which equilibrium or near-equilibrium states may be identified by direct visual inspection. This is of great help in many circumstances, for instance when studying the effect of new conditions, such as variations in the stress intensity or temperature, on the equilibrium state. Similar studies in a conventional simulation approach would, in general, require a large number of iterations of the run-simulation/analyse-data/run-simulation loop. But, probably, it is in the study of systems where dynamical effects are the main theme of study that interactive simulations show their best advantages. In the fracture examples above, for instance, the richness of details revealed in the run-time animation is far greater than what would be revealed by any conventional post-processing other than a full animation. Full animations, however, besides requiring huge amounts of stored data, as remarked above, do not provide any of the benefits of interactivity. As a last point, we would like to remark the easy with which one may identify and follow the unfolding of special phenomena, for instance, the emission and propagation of dislocations in the present implementation. This feature may be of great help when devising strategies and developing algorithms for the tracking of such special phenomena. We next give our conclusions.

Computer simulations have long been an important instrument for studying systems with a large number of degrees of freedom, such as the many-particle system considered in this article. The possibility of coupling such simulations with a graphical interface, for interactively inputting data and displaying results, opens new and fascinating possibilities. It has been our objective here to give an illustration of some of these, by exploring the specific examples of fracture and grain boundaries in a rather simplified two-

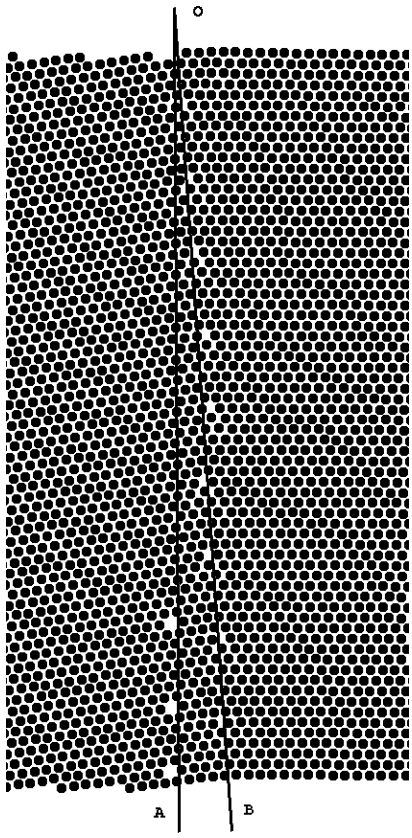

Fig. 6. Asymmetric tilt boundary. The parameters defining the simulated system are given in the text. Line OA indicates the initial position of the boundary, while line OB gives the final position.

dimensional model. A quick survey in the literature will show that the scope of applications of interactive simulations to physical systems is widening by the day. It already ranges from education [24] to advanced research [25].

We hope that experimenting with the program will succeed in conveying the basic ideas underlying an interactive approach to computer simulations of physical systems and in illustrating the usefulness of this concept when exploring the parameter space of such systems. Hopefully, it will stimulate either further improvement of the application described here or the design and implementation of new ones. We have greatly benefited from a previous work by D.C. Rapaport [5].

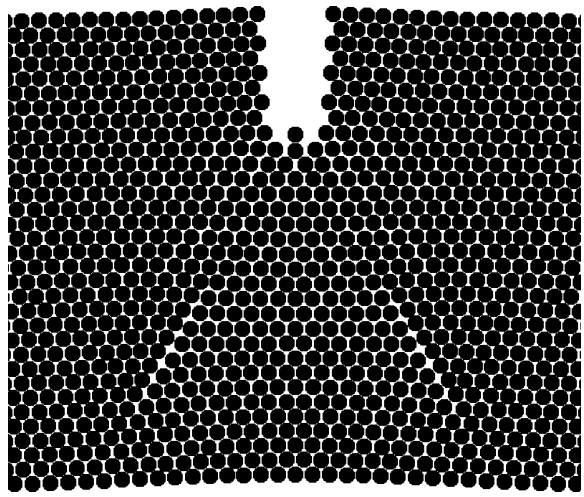

Fig. 7. Partial view of the graphics window in the ductile fracture example. Particle representation. Note the two dislocations emitted from the crack tip. Configuration after 6.27 picoseconds.

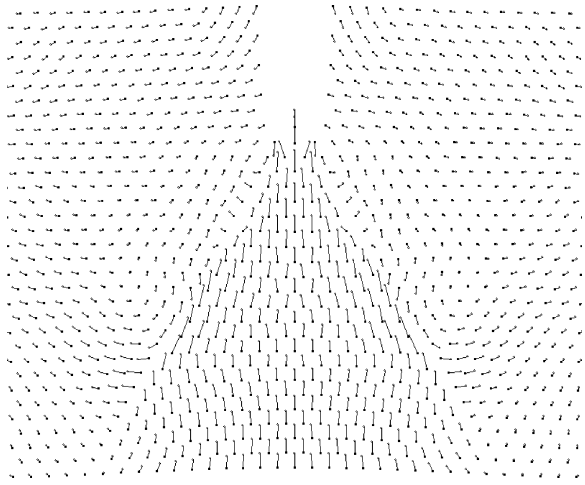

Fig. 8. Partial view of the graphics window in the ductile fracture example. The velocity field corresponding to the particles shown in Fig. 7 is displayed. Notice the configuration of the velocity field near the dislocations shown in that figure.

Acknowledgements

It is a pleasure to thank Prof. David Landau, from the University of Georgia, and Dr. Antti Kuronen, from the Laboratory of Computational Engineering/HUT, for helpful suggestions. This work has been partly supported by the Academy of Finland.

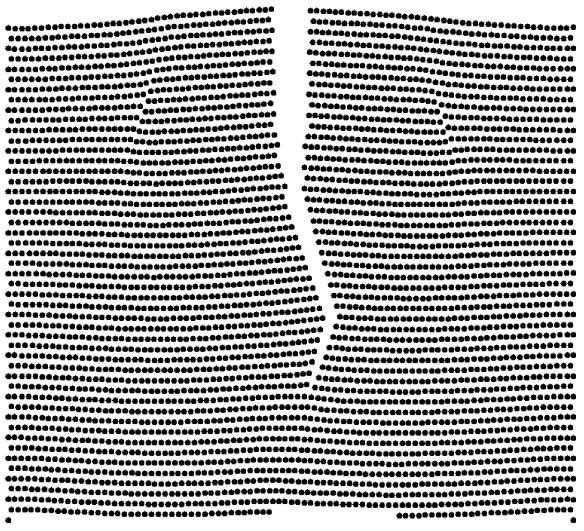

Fig. 9. Partial view of the graphics window in the brittle fracture example. Configuration after 7.58 picoseconds. Notice that the fracture propagates without any blunting.

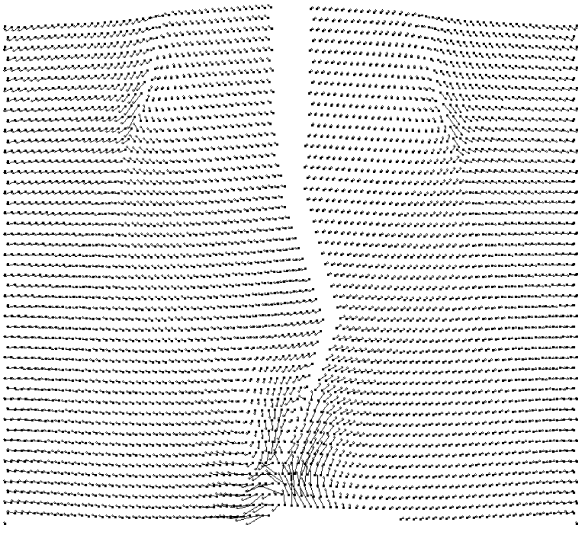

Fig. 10. Partial view of the graphics window in the brittle fracture example. The velocity field corresponding to the particles shown in Fig. 9 is displayed.

Appendix A. Two-dimensional dislocations

The two-dimensional counterpart of a 3D edge dislocation is the defect arising from the combination of one or more incomplete rows of atoms (lines) in a 2D structure. In a triangular lattice there are three families of most dense lines. Lines from one family are rotated 60° with respect to lines from the other two families. The most simple edge dislocation is formed by an incomplete line from any of the families. Combinations two at a time of incomplete lines from different families, however, give origin to a set of dislocations which have more stability than the above dislocation. Only these are seen in a triangular lattice, after equilibrium is reached. Both types of dislocations are illustrated in Fig. A.1. It may be noticed that there are three possible combinations of incomplete lines, giving origin to three different classes of dislocations. The dislocations in one class formed by two most dense incomplete lines can move in a conservative way [2] only along the direction of the third most dense line, which is not interrupted at the dislocation. This is also shown in Fig. A.1. These lines are the *slip lines* for the movement of stable dislocations in a triangular lattice.

Appendix B. Description of files

The program is distributed in nine files. An additional Makefile automates compiling and linking tasks. The file *graphics.c* holds the main loop as well as the definition of all graphical elements displayed in the windows ‘Parameters’ and ‘Fracture’. The associations of specific events (pushing of a button, change of a parameter, etc...) with routines (call-backs) are also defined in *graphics.c*. The file *callback.c* contains, as its name already suggests, the *call-back* routines called out from the *graphics.c* module. Files *simu.c* and *calc.c* contain the MD algorithm and the lattice generating routines (triangular lattices with edges arbitrarily oriented), respectively. The files with termination *h* hold definitions of macros, main types, variables and constants. Default values for most parameters are to be found in these files. In particular, units and main simulation parameters (integration time step, for example) are defined in the file *simu.h*.

[2] It is said that a dislocation moves in a conservative way when its movement takes place without the generation of any defects.

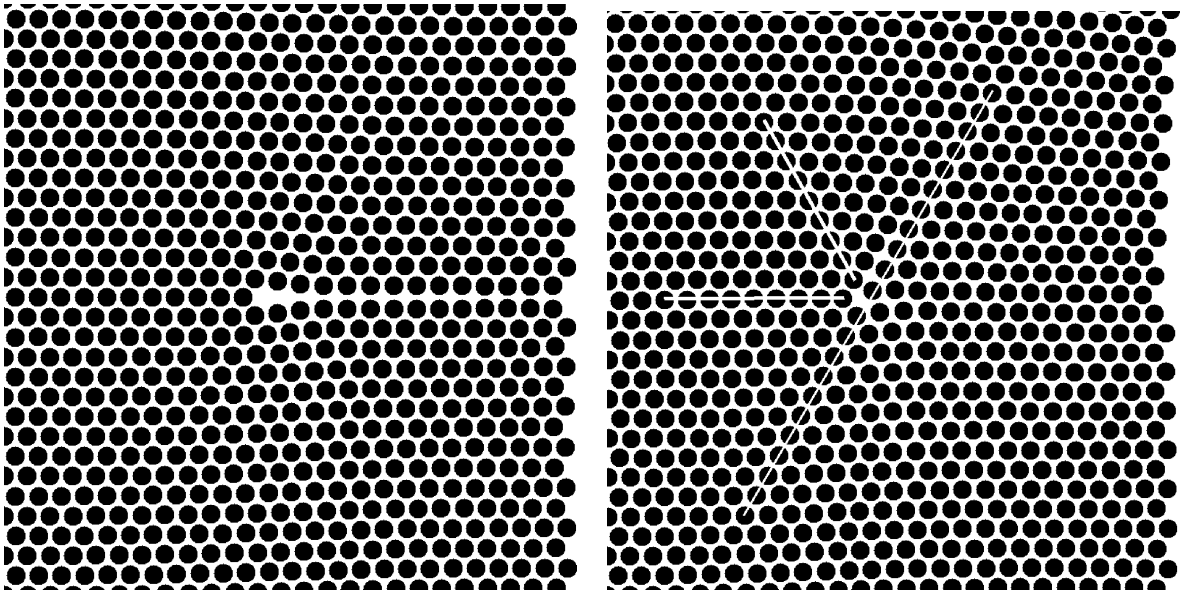

Fig. A.1. Dislocations in a triangular lattice. The figure on the left-hand side shows a unstable dislocation, while that on the right-hand side shows a stable one. Notice the existence of two extra *rows* in the latter. These have been marked with thick lines. The thin line indicates the line along which the dislocation can move (*slip line*).

6. Command line options

Some default values and definitions may be changed by adding options to the command line. The general format of the command line is

Fracture **-s** *size* **-bpp 16** **-c0** *colour* **-c1** *colour* **-c2** *colour*

where

-s *size* defines the *size* in pixels of the graphics area;

-bpp 16 enables the colour scheme described in Section 2.3. This feature is operative only in systems with a 16-bit or superior graphics card;

-c0 *colour* the background colour of the graphics area is set to *colour*;

-c1 *colour* colour of the atoms represented in the graphics area is set to *colour*;

-c2 *colour* colour of the atoms in the 'fixed borders' are set to *colour*.

The default size of the graphics area has been set for 14 inch monitors. For larger monitors, as is standard in most workstations, the option *-s 800* improves visualization effects.

References

[1] S.J. Zhou, D.L. Preston, P.S. Lomdhal, D.M. Beazley, Science 279 (1998) 1525.
[2] F. Abraham, Phys. Rev. Lett. 77 (1996) 869.
[3] S.J. Zhou, D.M. Beazley, P.S. Lomdhal, B.L. Holian, Phys. Rev. Lett. 78 (1997) 479.
[4] Theme edition: Advancing Interactive Visualization, IEEE Comput. Sci. & Eng. 4 (1996).
[5] D.C. Rapaport, Comput. Phys. 11 (1997) 337.
[6] V.M. Fernandez, D. Silver, N.J. Zabusky, Comput. Phys. 10 (1996) 463.
[7] D. Levine, M. Facello, P. Hallstrom, G. Reeder, B. Walenz, F. Stevens, IEEE Comput. Sci. & Eng. 4 (1997) 55.
[8] A.P. Sutton, R.W. Balluffi, Interfaces in Crystalline Materials, Monographs on the Physics and Chemistry of Materials (Clarendon Press, Oxford, 1995).
[9] J.P. Hirth, J. Lothe, Theory of Dislocations (Wiley, New York, 1982).
[10] V.L. Fitch, D.R. Marlow, M.A.E. Dementi, eds., Critical Problems in Physics, Princeton Series in Physics (Princeton University Press, Princeton, NJ, 1997).

[11] B. Lawn, Fracture in Solids, Cambridge Monographs in Physics (Cambridge University Press, Cambridge, 1997).
[12] E. Sharon, S.P. Gross, J. Fineberg, Phys. Rev. Lett. 74 (1995) 5096.
[13] P. Heino, K. Kaski, Phys. Rev. B 54 (1996) 6150.
[14] S. Ramanathan, D.S. Fisher, Phys. Rev. B 58 (1998) 6026.
[15] S.J. Zhou, P.S. Lomdhal, R. Thomson, B.L. Holian, Phys. Rev. Lett. 76 (1996) 2318.
[16] B.L. Holian, R. Ravelo, Phys. Rev. B 51 (1995) 11275.
[17] D.C. Rapaport, The Art of Molecular Dynamics Simulation (Cambridge Univ. Press, Cambridge, 1995).
[18] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids (Oxford University Press, Oxford, 1987).
[19] D. Heller, P.M. Ferguson, Motif Programming Manual (O'Reilly & Associates, USA, 1994).
[20] E. Cutler, D. Gilly, T. O'Reilly, The X Window System in a Nutshell (O'Reilly & Associates, USA, 1992).
[21] D. Flanagan, Motif Tools (O'Reilly & Associates, USA, 1994).
[22] D. Greenspan, Particle Modeling (Birkhäuser, Boston, 1997).
[23] D.R. Lide, Handbook of Chemistry and Physics (CRC Press, USA, 1994).
[24] R.H. Silsbee, Jörg Dräger, Simulations for Solid State Physics (Cambridge University Press, UK, 1997); see also http://www.ph.biu.ac.il/∼rapaport.
[25] W.-F. Ihlenfeldt, J. Mol. Model. 3 (1997) 386.