ELSEVIER Computational Materials Science 12 (1998) 278–287

COMPUTATIONAL MATERIALS SCIENCE

Tight-binding molecular dynamics simulations

Luciano Colombo [1]

Dipartimento di Scienza dei Materiali, Università degli Studi di Milano, via Emanueli 15, I-20126 Milano, Italy

Accepted 15 June 1998

Abstract

We present the tight-binding molecular dynamics (TBMD) scheme and describe its numerical implementation in a serial FORTRAN-77 code. We discuss how to organize a typical simulation and to control the I/O. An analysis of the computational workload is also presented and discussed. © 1998 Elsevier Science B.V. All rights reserved.

1. Introduction

Tight-binding molecular dynamics (TBMD) is a computational tool designed to run finite-temperature molecular dynamics (MD) simulations within the semi-empirical tight-binding (TB) scheme. The electronic structure of the simulated system is calculated by a TB Hamiltonian so that the quantum mechanical many-body nature of the interatomic forces is naturally taken into account.

Thanks to the semi-empirical character of the present TBMD tool, the resulting computational workload is comparatively small and allows for relatively large-scale simulations.

In Section 2 we present the basic theory of TB formalism while in Section 3 we introduce the TBMD FORTRAN-77 code. We do not discuss the MD scheme and refer to the book by Allen and Tildesley [1] for an excellent introduction to molecular dynamics. In Section 4 we show how to select a desired simulation and, finally, in Section 5 we give more details about the present implementation of TBMD.

2. The basic formalism

2.1. Tight-binding formalism for total-energy calculations

The Hamiltonian of a system of ion cores and valence electrons can be written in the adiabatic approximation as:

$$H_{\mathrm{tot}} = T_{\mathrm{i}} + T_{\mathrm{e}} + U_{\mathrm{ee}} + U_{\mathrm{ei}} + U_{\mathrm{ii}}, \tag{1}$$

where $T_{\mathrm{i,e}}$ is the kinetic energy of ions and electrons, while the electron–electron, electron–ion, and ion–ion interactions are given by $U_{\mathrm{ee}}, U_{\mathrm{ei}}, U_{\mathrm{ii}}$, respectively. In the spirit of the one-electron picture, [2] the many-body Hamiltonian given in Eq. (1) is reduced to the problem of one particle (electron) moving in an average field due to the other electrons and to the ions. Let H be the reduced one-electron Hamiltonian and $|\Psi_n\rangle$ its nth eigenfunction such that

$$H|\Psi_n\rangle = \epsilon_n|\Psi_n\rangle, \tag{2}$$

[1] Tel.: +39 2 66174218; fax: +39 2 66174403; e-mail: luciano.colombo@mater.unimi.it.

where ϵ_n is the energy of the nth single-particle state. In the TB formulation, the single-particle wavefunctions $|\Psi_n\rangle$ are cast in the form of a linear combination of atomic orbitals $|\phi_{l\alpha}\rangle$ [2]:

$$|\Psi_n\rangle = \sum_{l\alpha} c^n_{l\alpha} |\phi_{l\alpha}\rangle \tag{3}$$

where l is the quantum number index and α labels the ions. In general, the $|\phi_{l\alpha}\rangle$ basis set is not orthogonal. The use of a non-orthogonal basis set, even if in principle possible, is not numerically convenient in most cases. As a matter of fact, the evaluation of overlap integrals $\int \phi^*_{l'\beta}\phi_{l\alpha}\mathrm{d}\vec{r}$ is needed at each time step of the simulation and the computational workload increases accordingly. However, it is possible to orthogonalize the atomic orbitals $|\phi_{l\alpha}\rangle$ in such a way that the new basis set functions still maintain their symmetry properties. Such an operation defines a new orthogonal basis set whose functions $\{\varphi_{l\alpha}\}$ are called Löwdin orbitals [2]. The Schrödinger equation for the single-particle states is finally reduced to

$$\sum_{l'\beta} \Big(\langle \varphi_{l'\beta}|H|\varphi_{l\alpha}\rangle - \epsilon_n \delta_{ll'}\delta_{\alpha\beta} \Big) c^n_{l'\beta} = 0. \tag{4}$$

Within a semi-empirical TB, the matrix elements of the Hamiltonian entering in Eq. (4) are evaluated by fitting a suitable database obtained either from experiments or by first principles calculation. Typically, the fitting is operated onto the electronic energy bands [2,3]. In order to keep the number of TB parameters as low as possible, a number of approximations are here introduced: (i) a minimal basis set is selected (i.e. just a sp^3 basis set is used for silicon and carbon) (ii) only two-center integrals are taken into account; (iii) short-range interactions are assumed.

Once the single-particle energies are known by solving the secular Eq. (4), the total energy E_{tot} of a system of ion cores and valence electrons can be written as:

$$E_{\mathrm{tot}} = \sum_n \epsilon_n f(\epsilon_n, T) + U_{\mathrm{ii}} - U_{\mathrm{ee}} \tag{5}$$

where $f(\epsilon_n, T)$ is the Fermi–Dirac distribution function and the $-U_{\mathrm{ee}}$ contribution corrects the double counting of the electron–electron interactions in the first term. The sum over all the single-particle energies is commonly named band structure energy E_{bs}. The last two terms appearing in Eq. (5) are usually grouped into an effective repulsive potential $U_{\mathrm{rep}} = U_{\mathrm{ii}} - U_{\mathrm{ee}}$ which, in turn, can be expressed as a sum of suitable two-body potentials:

$$U_{\mathrm{rep}} = U_{\mathrm{ii}} - U_{\mathrm{ee}} = \sum_{\alpha,\beta>\alpha} \Phi(r_{\alpha\beta}) \tag{6}$$

where $r_{\alpha\beta}$ is the distance between the αth and βth atom. U_{rep} contains the effects of both the overlap interaction originated by the non-orthogonality of the basis orbitals (neglected in E_{bs}) and the possible charge transfer effects (that are not self-consistently included in the TB scheme). Here we adopted the form of $\Phi(r_{\alpha\beta})$ given in Ref. [3].

2.2. Tight-binding molecular dynamics

During an MD run we must calculate TB matrix elements (see Eq. (4)) between atoms which are not at their equilibrium positions. Consistently, we need a scaling law $g(r_{\alpha\beta})$ for the variation of the fitted $\langle \varphi_{l'\beta}|H|\varphi_{l\alpha}\rangle$ against $r_{\alpha\beta}$. It is usually determined, along with $\Phi(r_{\alpha\beta})$, by fitting the zero temperature cohesive-energy curves of the system of interest, calculated from first principle for different phases [3]. It is worth noticing that all the parameters entering into the present TBMD scheme (TB hopping integrals, repulsive potential and scaling law) result definitely fixed at this level: no further adjustments of the parameters will be operated during the simulation.

The forces $\vec{f}_\alpha$ ($\alpha = 1, 2, \ldots, N_{\mathrm{at}}$) needed to move atoms can now be straightforwardly evaluated from the TBMD Hamiltonian H_{TBMD}

$$H_{\mathrm{TBMD}} = \sum_\alpha \frac{p^2_\alpha}{2m_\alpha} + \sum_n \epsilon_n f(\epsilon_n, T) + U_{\mathrm{rep}} \tag{7}$$

and are given by

$$\vec{f}_\alpha = -\sum_n \left\langle \Psi_n \left| \frac{\partial H}{\partial \vec{r}_\alpha} \right| \Psi_n \right\rangle f(\epsilon_n, T) - \frac{\partial U_{\mathrm{rep}}}{\partial \vec{r}_\alpha}. \tag{8}$$

The second term on the right-hand side of the above equation is trivially calculated and it is not computer demanding. In fact, the potential $\Phi(r_{\alpha\beta})$ is known analytically and it is short ranged [3]. On the other hand, the first term contains the

Hellmann–Feynman contribution to the total force and can be calculated as:

$$\sum_n \left\langle \Psi_n \left| \frac{\partial H}{\partial \vec{r}_\alpha} \right| \Psi_n \right\rangle f(\epsilon_n, T) =$$

$$-2\sum_n f(\epsilon_n, T) \sum_{l\gamma} \sum_{l'\beta} c^n_{l'\beta} \frac{\partial H_{l'\beta,l\gamma}(r_{\beta\gamma})}{\partial \vec{r}_\alpha} c^n_{l\gamma}, \tag{9}$$

where we have put

$$H_{l'\beta,l\gamma}(r_{\beta\gamma}) = g(r_{\beta\gamma}) \langle \varphi_{l'\beta} | H | \varphi_{l\gamma} \rangle |_{r_{\beta\gamma}=r_0}. \tag{10}$$

It is clear from Eqs. (5) and (9) that we need the full spectrum of eigenvalues $\{\epsilon_n\}$ and eigenvectors $\{c^n_{l\alpha}\}$ of the TB Hamiltonian matrix in order to calculate E_{bs} and Hellmann–Feynman forces, respectively.

Finally, we remark that in the present release of the TBMD code, the zero-temperature Fermi–Dirac distribution function is assumed everywhere.

3. The source code TBMD

3.1. Overview of the code

The full TBMD library consists in the following parts:

- the main program,
- 14 subroutines,
- one function,
- 17 include files,
- one input file,

where: the main program, the 14 subroutines and the function are contained in just one source file named **tbmd.f**. The input file is named **tbmd.in** (described in Section 3.5) and the 17 include files are named as follows (described in Section 3.4):

acc.inc dist.inc eigen.inc erg.inc fhf.inc frep.inc ftot.inc func.inc latt.inc lst.inc param.inc pos.inc potz.inc simul.inc start.inc tb.inc vel.inc

The programming language is FORTRAN-77. We have put special care in order to write a portable source code. The resulting code was benchmarked on the following platforms and no bugs have been found: DIGITAL VAX, VAXstations and DECstations; SUN, HP and IBM RISC workstations; CONVEX C3820; CRAY Y-MP; NEC SX-3 and SX-4.

Finally, one subroutine for matrix diagonalization is needed. In the present release of the TBMD code we make use of the routine SSYEV from LAPACK. It computes the full spectrum of eigen-values/-vectors of a real, symmetric matrix. We refer to LAPACK manual for more information about it.

3.2. Description of the operations

In the following we describe in detail the operations performed by the main program, the subroutines and the function.

main program (named: **program TBMD**)
It reads the input data and writes all the output information. It initializes the simulation and performs the MD loop over the selected number of time-steps. It controls the flow of data in the code. It selects the temperature by rescaling atomic velocities at each time-step in a constant-temperature MD run.

subroutine bondlength
It computes the bond-length distribution function.

subroutine correlate
It computes the pair correlation function and the radial distribution function [1].

subroutine crystal
It defines the tight-binding, repulsive potential and lattice parameters.

subroutine dosenergy
It computes the total electronic density of states [2].

subroutine feynman
It computes the Hellmann–Feynman forces (Eqs. (8)–(10)).

subroutine htb
It computes and diagonalizes the tight-binding matrix (Eq. (4)); it computes the band structure energy (Eq. (5)).

subroutine init-pos
It gives the initial positions of the atoms in the simulation box (see also Section 4).

subroutine init-vel
It gives the initial velocities of the atoms in the simulation box.

subroutine md

It performs the main molecular dynamics loop. The atomic positions, velocities and accelerations are updated according to the velocity Verlet algorithm [1]. The Verlet list for the interacting particles and periodic boundary conditions is used [1].

subroutine meandisp

It computes the atomic mean square displacement [1].

subroutine nearangle

It computes bond-angle distribution function and atomic coordination number [1].

subroutine pressure

It computes pressure according to the virial theorem [1].

subroutine repuls

It computes repulsive energy (Eq. (6)) and forces (Eq. (8)).

subroutine temper

It computes temperature [1].

Finally, the function named **RAN1** generates a random number in the interval]0,1[and it is called by the subroutines **init-pos** and **init-vel**. The function **RAN1** has been taken from the book "Numerical Recipes – The Art of Scientific Computing" by W.H. Press, B.P. Flannery, A.A. Teukilsky and W.T. Vetterling (Cambridge University Press, 1986).

3.3. List of main variables

We list the main variables used in the **tbmd.f** file. Secondary and local variables are described in the source code. We quote also the reduced units adopted to measure different quantities.

itype	select the material (1 = carbon; 2 = silicon)
ncell	number of replicas of the conventional diamond cell (8-atom cell) along the x, y, z directions. It defines the size of the simulation box (see subroutine crystal and Section 4)
n	number of atoms in the simulation box. It is: $n = 8 \times (ncell)^3$
x, y, z	x, y, z components of the atomic position vector (reduced units: Å)
vx, v y, vz	x, y, z components of the atomic velocity (reduced units: $\sqrt{\mathrm{eV/amu}}$ or 9.822694×10^{13} Å/s)
ax, ay, az	x, y, z components of the acceleration (reduced units: eV/Å/amu or 9.648532×10^{27} Å/s^2)
ax/y/zold	x, y, z components of the atomic acceleration at the previous step (reduced units: eV/Å/amu)
xx, y y, zz	x, y, z components of the relative distance vector between two atoms (ex.: $xx(i,j) = x(j) - x(i)$) (reduced units: Å)
rr2	squared relative distance between two atoms (reduced units: Å^2)
tempi	initial temperature (K)
tempf	final temperature (K)
temp	instantaneous temperature (K)
press	istantaneous pressure (reduced units: eV/Å^3 or 1.602177×10^{11} Pa)
cell	size of the simulation box (see subroutine crystal and Section 4) (reduced units: Å)
eval	eigenvalues of tight-binding Hamiltonian matrix
utot	repulsive potential energy per atom (eV)
ebstot	band-structure energy per atom (eV)
kintot	kinetic energy per atom (eV)
energy	total energy per atom (eV)
frepx/y/z	x, y, z components of the repulsive force (reduced units: eV/Åor 1.602177×10^{-4} dyne)
fhfx/y/z	x, y, z components of the Hellmann–Feynman force (reduced units: eV/Å)
ftotx/y/z	x, y, z components of the total force (reduced units: eV/Å)
nstep	number of time-steps of the simulation
dt	time step (reduced units: 1 time-step = $1.018050697 \times 1\ 0^{-14}$)
iseedp	seed of random generation (see subroutine init-pos)
iseedv	seed of random generation (see subroutine init-pos)

cut	cutoff for interactions (reduced units: Å)
bcut	cutoff for the computation of the bond-angle distribution function and atomic coordination number (see subroutine nearangle) (reduced units: Å)

3.4. The include files

The above include files are intended to declare variables and put them in suitable common blocks. In particular:

ACC.INC defines variables *ax/y/zold*; common block ACCELER

DIST.INC define variables *xx/yy/zz/rr2*; common block DISTANCE

EIGEN.INC defines variables *eval*, *h* and *occup*; common block EIGENV

ERG.INC defines variables *kintot*, *ebstot*, *utot*, *energy*; common block ENERGIES

FHF.INC defines variables *fhfx/y/z;* common block HELLFEYN

FREP.INC defines variables *frepx/y/z;* common block REPULSIVE

FTOT.INC defines variables *ftotx/y/z;* common block FORCETOT

FUNC.INC defines temp and press; common block FUNZ

LATT.INC defines variables mass, *r0*, *l0*, *r02*, *kboltz*; common blocks LATTICE and CHEMICAL

LST.INC defines the variables *list*, *list1*, *nlst*, *nlst1*; common block VERLET

PARAM.INC defines variables *ncell*, *n*, *nn4* and *n4*

POS.INC defines variables *x/y/z*(*n*), *cell*, *cell2*; common block POSITION

POTZ.INC defines the repulsive potential parameters (i.e. *phi*, *rc*, *enc*, *em*,...); common block GOODWIN

SIMUL.INC defines variables *dt*, *dtsqr*, *dthalf*, *cutoff*, *bcutoff*; common block EXPER

START.INC defines variables *temp0*, *shift*, *nstep*, *ntemp*; common block INITIAL

TB.INC defines variables *ess*, *epp*, *sss*, *sps*, *pps*, *ppp*; common block DATATB

VEL.INC defines variables *vx/y/z*; common block VELOCITY

3.5. The input file tbmd.in

The input file **tbmd.in** allows to select a desired TBMD simulation. Through **tbmd.in** we can:

- select a constant-energy (NVE) or constant-temperature (NVT) simulation
- choose if starting a new simulation or continuing from the configuration generated by a previous one
- choose the number of time-steps of the simulations
- select whether (and every how many time-steps) saving atomic positions (trajectories) and velocities for post-processing
- select the frequency for saving intermediate configuration for restarting (this option is intended to avoid the loss of any information due to possible system crashes)
- select which physical observables must be calculated
- choose the time-step
- choose the cutoff
- choose density of the system
- define inital temperature of the simulation
- define final temperature of the simulation (if an NVT simulation has been selected).

If we like to start from the configuration generated by a previous simulation, a data file generated by the present TBMD program must be present in the working directory. The logical name for the character variable corresponding to that file is:

finp: input filename (containing atomic positions, velocities and accelerations of all particles in the simulation box, as calculated in the last time-step of a previous simulation)

The **tbmd.in** file has been created in order to be very user-friendly. It consists of a sequence of character strings (comments) and numbers (or filenames). You must simply edit it and type the values you like for the variables. In the following we report a typical example of a **tbmd.in** file.

..... file **tbmd.in** starts .....

'Select microcanonical (1) or canonical (any integer) ensemble'

2

```
'New simulation (0) or continue a previous one (1)'
1
'Input filename containing starting atomic configuration (max 15 c)'
'config1'
'Number of time-steps in the simulation '
10 000
'First step for saving atomic positions'
1
'Last step for saving atomic positions'
5000
'Saving atomic positions every... time-steps'
2
'Filename for atomic positions (max 15 c)'
'positions'
'First step for saving atomic velocities'
39 501
'Last step for saving atomic velocities'
35 902
'Saving atomic velocities every... time-steps'
35 900
'Filename for atomic velocities (max 15 c)'
'vel'
'Write temperature and pressure every... time-steps'
10
'Filename for temperature and pressure (max 15 c)'
'temperature'
'Write kinetic, repulsive, band-structure and total energy every... time-steps'
10
'Filename for energy (max 15 c)'
'energy'
'Save atomic configuration for restarting every... time-steps'
1000
'Filename containing atomic configuration (max 15 c)'
'config2'
'Compute mean square displacement? (y=1/n=0)'
0
'Initial step for computing mean square displacement'
1000
'Last step for computing mean square displacement'
2000
'Filename for mean square displacement (max 15 c)'
'disp'
'Compute pair correlation function? (y=1/n=0)'
1
'Initial step for computing pair correlation function'
5000
'Last step for computing pair correlation function'
10 000
'Filename for pair correlation function (max 15 c)'
'correlation'
'Compute bond-angle distribution and atomic coordination number? (y=1/n=0)'
1
'Initial step for computing bond-angle distr. and atomic coord. number'
5000
'Last step for computing bond-angle distr. and atomic coord. number'
10 000
'Filename for bond angle distribution (max 15 c)'
'bonddistr'
'Filename for atomic coordination number (max 15 c)'
'coordination'
'Compute bond-length distribution? (y=1/n=0)'
0
'Initial step for computing bond-length distr. '
1000
'Last step for computing bond-length distr. '
2000
'Filename for bond-length distribution (max 15 c)'
'lengthdistr'
'Compute electronic density of states? (y=1/n=0)'
0
'Initial step for computing electronic density of states'
```

1000
'Last step for computing electronic density of states'
2000
'Filename for electronic density of states (max 15 c)'
'dos'
'itype: 1 = carbon; 2 = silicon; other = not implemented'
2
'Select density (give the length of conventional diamond cell)'
'(Example: diamond *l*0 = 3.547999847; silicon: *l*0 = 5.45099945)'
5.45099945*d*0
'Time-step (in reduced units: 1.018*d*-14 s)'
0.1*d*0
'Interaction cutoff: C = 2.6*d*0 Si = 4.15*d*0 (see reference papers)'
4.2*d*0
'Cutoff for computing bond angle distr. and atomic coord. number (Angstrom) '
3.1*d*0
'Final temperature of the simulation'
300.0*d*0
'Initial shift from perfect lattice positions (% of 1nn distance in diamond lattice)'
0.01*d*0
'iseed for random generation of positions'
-124
'iseed for random generation of velocities'
-135
'Initial temperature of the simulation'
300.*d*0
..... file **tbmd.in** ends .....

The above example is relative to a simulation where:

– an NVT simulation over 10 000 time-steps is started from the file **config1**
– the atomic positions are stored every two time-steps in the file **positions** starting from the first time-step up to the 5000th one
– atomic velocities are NOT stored
– instantaneous temperature and energy are saved every 10 time-steps in the files **temperature** and **energy**, respectively
– intermediate and final configurations are saved every 1000 time-steps in the file **config2**
– the atomic mean square displacement is not calculated
– the pair correlation function is calculated from the 5000th time-step up to the 10 000th one and saved in file **correlation**
– the bond-angle distribution and atomic coordination are calculated from the 5000th time-step up to the 10 000th one and saved in files **bonddistr** and **coordination**, respectively
– the bond-length distribution function is not calculated
– the electronic density of states is not calculated
– the density corresponds to crystalline silicon
– the time-step is 10^{-15} s
– the interaction cutoff is set equal to 4.2 Å
– the bond cutoff is set equal to 3.1 Å
– initial and final temperatures are equal to 300 K.
– since we start from file **config1**, the random shift of ideal crystalline positions is not operated and the seeds for random generations are not used here.

The above example corresponds to a simulation where the system is NOT heated or cooled. If *tempi* and *tempf* are different and the NVT scheme is selected, then it is possible to heat or cool the system. The heating/cooling rate is easily calculated from the difference *tempf*-*tempi* and the number of time steps *nstep*. If the NVE scheme is selected, no control on the temperature is possible.

Further details about how select and run a TBMD simulations are given in Section 4. In particular, we discuss in Section 4 how to select the number of particles present in the simulation box.

3.6. The output

The output information of a TBMD run is stored in a number of different files. They are:

flast	output filename (containing atomic positions, velocities and accelerations of all particles in the simulation box, as calculated in the last time-step of the present simulation). It is used for both on-the-fly and final saving of the atomic configuration.

fcorr	filename for pair correlation function and radial distribution function (in the order: distance, pair correlation function, radial distribution function)
fbond	filename for bond angle distribution (in the order: angle, bond angle distribution function)
flength	filename for the bond length distribution
fnear	filename for atomic coordination number (in the order: coordination number, % of atoms having that coordination)
fdisp	filename for atomic mean square displacement (in the order: time-step number, mean square displacement)
feval	filename for total electronic density of states (in the order: energy, density of states)
ftp	filename for temperature and pressure (in the order: time-step number, reference temperature, instantaneous temperature, instantaneous pressure)
ferg	filename for kinetic, potential, band-structure and total energy (in the following order: time-step number, kinetic energy, repulsive energy, band-structure energy, total energy)
fpos	filename for atomic trajectories (unformatted)
fvel	filename for atomic velocities (unformatted)

Files **fpos** and **fvel** are written as unformatted files, according to the following statements:

```
    open(unit = 97,file = fpos,status = 'un-
    known',form = 'unformatted')
    open(unit = 96,file = fvel,status = 'un-
    known',form = 'unformatted')
    write(97) n, nwrpos, nstep
    write(96) n, nwrvel, nstep
    do 150 i = 1, n
    write(97) x(i), y(i), z(i)
150 continue
    do 151 i = 1, n
    write(96) vx(i), vy(i), vz(i)
151 continue
```

where positions and velocities are saved every *nwrpos* and *nwrvel* time-steps, respectively (see Section 3.5).

Note that we have above reported the symbolic name given in the code to the character variable corresponding to any file. The actual output filename can be assigned as discussed in the previous section.

3.7. Workload analysis

During a conventional TBMD simulation, most of the computational work is spent in calculating the eigenvalues $\{\epsilon_n\}$ and the eigenvectors $\{c^n_{\alpha\beta}\}$ of the TB Hamiltonian entering Eqs. (5) and (9), respectively. This procedure requires a standard diagonalization of a real, symmetric matrix which could be quite large for systems with $N_{\mathrm{at}} > 500$ (the TB basis set consists of four orbitals per atom). We observed that, typically, in a TBMD run more than 90% of the CPU time requested at each time-step is consumed for the matrix diagonalization. We found that this result is in general valid whatever computer (workstation or mainframe, scalar or vector machine) is used, the actual percentage depending on the specific mathematical library used. The resulting computational method scales as the cubic power of N_{at} as far as the requested CPU is considered ($O(N^3_{\mathrm{at}})$ method). Unfortunately, iterative diagonalization techniques are not useful in the present scheme. As a matter of fact, if Eqs. (5) and (9) are used to evaluate the total energy of the system and the interatomic forces, *all* the eigenvalues and *all* the eigenvectors of the TB Hamiltonian are needed. Even if we assume that the electron system has a zero temperature distribution function (this is a good approximation in most cases), we do need half of them and iterative diagonalizations are not convenient.

The conventional TBMD scheme as outlined above is, however, rather efficient. In the following, we report timing information about three benchmark calculations that were run on the vector NEC SX-4 computer available at the Swiss Center for Scientific Computing SCSC/CSCS (Manno, CH). For a simulation cell containing as many as 64, 216, 512, and 1000 atoms the code required was 0.378, 5.359, 52.725, and 351.939 CPU seconds per time-step, respectively. The operations have been more than 98% vectorized, while the maximum sustained speed of execution was observed to be 1.44 Gflops for the 1000-atom case. We remark that we used a small number of time-steps just for running the tests on the daytime queues with short CPU time-limit. When two tests with same number of atoms and different number of time-steps are executed, the resulting Mflops may differ by about 5%.

Finally, as for the memory request, it ranges from ∼6 Mbyte for the 64-atom case, to 280 Mbyte for 1000-atom cell.

4. Running a TBMD simulation

We still need one basic information: how to choose the number of atoms in the simulation box. This is possible through the include file **param.inc** which reads as follows:

```
c***************************************
c  RELEASE TBMD.2.0
c  Last revision: November 21, 1996
c***************************************
c Contact author: Luciano Colombo
c  e-mail: luciano.colombo@mater.unimi.it
c***************************************
c
c  PARAM.INC is an 'include file' which assigns the values to the
c  parameters ncell and nc
c
      parameter(ncell = 3,   n = 8*ncell**3,
     nn4 = 4*n*n, n4 = 4*n)
```

This is the only include file that we need to edit before compiling and executing the TBMD code. The method adopted to choose the number of atoms is based on the conventional cubic cell of the diamond-like lattice. Such a cell contains eight atoms and the simulation box is generated by repeating the conventional cell *ncell* times along the x, y and z directions. The resulting box will contain $(ncell)^3$ conventional cells and $n = 8 \times (ncell)^3$ atoms. Accordingly, by modifying the parameter *ncell* we can create larger and larger simulation boxes: $ncell = 1, 2, 3, 4, 5$ corresponds to 8, 64, 216, 512, and 1000 atoms, respectively. Any array or matrix defined in **tbmd.f** has dimensions given by either n or $n4$ or $nn4$.

In conclusion, to organize a simulation:
- select the proper number of atoms by editing the file **param.inc**
- compile the source code to generate the executable file
- update the **tbmd.in** file
- submit execution

Good luck and enjoy TBMD!

5. Final remarks

It should be clear from the above presentation of the TBMD code that the simulation box is fixed in volume and shape. The only modification to the box volume can be operated through the **tbmd.in** file by selecting a suitable density. This is done, however, once for all at the very beginning of the simulation. Moreover, constant-temperature (NVT) simulations are obtained by direct rescaling the atomic velocities at each time-step of the simulation. Both these features can be updated in a more advanced release of the code where constant-pressure and Nosè thermostat are introduced [1].

As for the accuracy, we have tested that the total energy is conserved in a microcanonical (NVE) simulation of arbitrarely long duration up to 10^{-4} eV/atom for crystalline silicon at room temperature. When testing energy conservation on disordered structures (like, for instance, amorphous silicon), a lower precision was observed. This is due to the behaviour of the repulsive potential and scaling laws for TB matrix elements close to the cutoff distance and to the possible presence of atoms in that region. Actually, no cutoff functions were used to switch off smoothly the potential to zero outside the sphere of interaction. One can improve this feature following the suggestions of Ref. [3].

The reliability of the TBMD framework and, in particular, of the present implementation has been proved by a number of successful calculations. We refer to Ref. [4] for further details.

References

[1] M.P. Allen, D.J. Tildesley (Eds.), Computer Simulation in Chemical Physics, Kluwer Academic Publishers, Amsterdam, 1993.

[2] W.A. Harrison, Electronic Structure and the Properties of Solids, Dover Publications, New York, 1989; J.C. Slater, G.F. Koster, Phys. Rev. 94 (1954) 1498.

[3] C.H. Xu, C.Z. Wang, C.T. Chan, K.M. Ho, J. Phys. Condens. Matter 4 (1992) 6047; I. Kwon, R. Biswas, C.Z. Wang, K.M. Ho, C.M. Soukolis, Phys. Rev. B 49 (1994) 7242.

[4] L. Colombo, in: D. Stauffer (Ed.), Annual Reviews of Computational Physics, vol. IV, World Scientific, Singapore, 1996.