Software azTotMD

(parallel version for videocard with CUDA technology)

User manual

The azTotMD computer program (hereinafter referred to as the Program) is intended for computer modeling by classical molecular dynamics (MD) and non-constant force field molecular dynamics.

System requirements

Operating system MS Windows 10 (not tested with other operating systems).

NVidia video card with Computation Capability> 2.2.

1 The program capabilities

Velocity Verlet integrator.

Microcanonical (NVE) and Canonical (NVT) Ensembles.

Rectangular periodic conditions.

Thermostats:

- Without thermostating (NVE ensemble);
- Nose-Hoover;
- Radiative.

Accounting for electrostatics:

- Naive way (Coulomb's law with cutoff);
- Ewald summation;
- Method of Fennel and Geselter [Fenn].

Short-range intermolecular pair potentials.

Intramolecular potentials:

- Valent bonds;
- Valent angles.

External electric field.

Non-constant force field:

- Electron transfer;
- Dynamic formation / removal of valent bonds;
- Hydrogen bonds (including bifurcate ones);
- Dynamic formation / removal of valent angles.

2 Working with the Program

The Program has no graphical user interface (GUI). Information input / output is carried out through files. The program reads information from 6 files, 4 of which are mandatory. Input files:

- atoms.xyz (mandatory) – box size and particle coordinates.
- field.txt (mandatory) – particle characteristics, force field.
- control.txt (mandatory) – general directives.
- cuda.txt (mandatory) – directives related to the video card.
- bonds.txt (optional) – a list of valent bonds.
- angles.txt (optional) – a list of valent angles.

2.1 The atoms.xyz file

- The first line is the number of particles.
- The second line is the boxing characteristic: first comes the number 1, and then the dimensions along the x, y and z axes (in angstroms, Å).
- Subsequent lines - type and coordinates of particles (x y z) in angstroms (Å).

All particle types must be defined in the field.txt file. Hereinafter, it is assumed that any number of spaces and/or tabs and/or line breaks are used as delimiters for values. An example of the atoms.xyz file (only the first few lines are shown):

```
3361
1 36.500000 36.500000 36.500000
V     2.935293   1.855377   0.381776
Od    2.663695   2.704217   1.708393
V     2.423310   3.419446   3.390218
Od    1.036296   1.359655   0.216446
Oc    3.856887   0.492592   1.200182
Oc    2.639205   1.741781   4.156950
```

2.2 The field.txt file

2.2.1 Definition of particle types

Specify the **spec** keyword and the number of particle types. Then, line by line, enter the characteristics of each type of particles: name (arbitrary, up to 7 characters, excluding whitespace), belonging to the "nuclei" (arbitrary, up to 7 non-blank characters), mass (in amu), charge (in proton charges), "own energy" (in eV, needed in very rare cases, you can specify 0). An example of a spec section:

```
spec 6
O2-   O     16.0  -2.0  0.0
Od    O     16.0  -1.2  0.0
P5    P     31.0  5.0   0.0
```

```
P3    P     31.0  4.2   0.0
V5    V     51.0  5.0   0.0
V3    V     51.0  4.2   0.0
```

2.2.2 Setting intermolecular pair potentials

Specify the keyword **vdw** and the number of pair potentials. Next, line by line, enter the characteristics of each pair potential: types of particles between which this pair potential acts; a keyword denoting a functional form of a pair potential; cutoff radius (in Å); parameters of paired potentials. Supported forms of pair potentials and the corresponding number of parameters are shown in Table 1. All values are indicated in Å, eV and their derived units. All particle types must be defined in the spec section. Example of a vdw section:

```
vdw 3
V3    Ot    elin   6.0   80.0      0.3    0.75
V3    Od    bmhs   1.0   2.68839   3.27417   1.78352   18.01372   -1.06927
V4    Od    buck   1.2   1290.56   0.34039   0.0
```

Table 1. Keywords and corresponding functional forms of pair potentials, number of parameters.

Keyword Potential name	**Functional form**	**Sequence of parameters' specifying**
lnjs Lennard-Jones potential	$U = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$	ε, σ
buck Buckingham potential	$U = A\exp(-r/\rho) - \frac{C}{r^6}$	A, ρ, C
bmhs Born-Mayer-Huggins potential	$U = A\exp[B(\sigma - r)] - \frac{C}{r^6} - \frac{D}{r^8}$	A, B, σ, C, D
p746 potential from paper [Staff]	$U = \frac{A}{r^7} - \frac{B}{r^4} - \frac{C}{r^6}$	A, B, C
elin exponent + linear function	$U = A\exp(-x/\rho) + Cr$	A, ρ, C
einv exponent - invert function	$U = A\exp(-x/\rho) - C/r$	A, ρ, C

2.2.3 Setting donor-acceptor properties. Required to implement the electron transfer option. Specify the keyword **red-ox** and the number of redox sequences. Next, enter the sequences line by line: the number of particles in the sequence and their types, starting from the highest

oxidation degree in order. The oxidation degrees of each next particle in the sequence must differ by 1, otherwise split the sequence into several ones. All particle types must be defined in the spec section. An example of a red-ox section:

```
red-ox 4
4       V5      V4      V3      V2
2       Cl0     Cl-
2       Mn7     Mn6
3       Mn4     Mn3     Mn2
```

2.2.4 Setting the types of valent bonds. To determine the potentials of valence bonds, specify the keyword **bonds** and the number of bond types. Next, write down the properties of each bond type line by line. The string describing the valent bond potential contains:

- serial number (indicate numbers starting from one and further in order, the Program numbers the bond types in order, ignoring specific numbers, they are intended for user convenience);
- a pair of particles between which the bond potential acts;
- the keyword denoting the functional form of the potential and the values of the potential parameters, table 2;
- a keyword that denotes the effect when the bond is shortened: con / mut (con - the bond does not change during compression, mut - the bond changes when shortened, in this case, further you need to specify the distance (in Å) below which the bond changes its type and the number of the new bond type);
- a keyword that denotes the effect when the bond is stretched: con / br / mut (con - the bond does not change when stretched; br - the bond is broken by stretching, after br indicate the break distance (in Å) and the new particles' types, into which current particles types transform after the bond breaking (in the order corresponding to their specification in the bond declaration); mut - the bond turns into another one when stretched, indicate the transformation distance and the number of the new bond type).

All particle types must be defined in the spec section. An example of the bonds section.

```
bonds 4
1       V3      Od      harm    50.0    1.62    con     con
2       V       Od      mors    10.0    1.4959  1.584   10.0    con     mut     1.8     3
3       V+      Ot      mors    6.496   1.4959  1.791   6.496   mut     1.8     2       con
4       Hh      Ot      mors    0.22    1.0     2.04    0.22    con     br      2.2     H       Ot
```

In this example, the bond of the 1st type remains constant, the 2nd one turns into a bond of the 3rd type with an increase in length to 1.8 Å (in this case, the bound particles V and Od turn into V + and Ot, respectively) and, conversely, the bond of the third type turns into the 2nd with a decrease in its length below 1.8 Å, with a corresponding change in the types of bonds. The type 4 bond breaks when its length reaches 2.2 Å with the transformation of the constituent particles (Hh and Ot into H and Ot, respectively, i.e., the second particle does not change the type). The bond of the 1st type has a functional form of the harmonic potential, the others are in the form of the Morse potential.

Table 2. Keywords and corresponding functional forms of valent bond potentials, number of parameters.

Keyword Potential name	**Functional form**	**Sequence of parameters' specifying**
harm harmonic potential	$U = ½k\,(r - r_0)^2$	k, r_0
mors Morse potential*	$U = D[1 - \exp(-\alpha(r - r_0))]^2 - C$	D, α, r_0, C
pdn potential from Pedone papers **	$U = D[1 - \exp(-\alpha(r - r_0))]^2 - C - E/r^{12}$	D, α, r_0, C, E
buck Buckingham potential	$U = A\exp(-r/\rho) - \frac{C}{r^6}$	A, ρ, C
e612	$U = A\,\exp(-x/\rho) - C/r^6 - D/r^8 - E/r^{12}$	A, ρ, C, D, E

* the original Morse potential has 3 parameters, $C = D$

** in original Pedone potential $C = D$

2.2.5 Setting hydrogen bonds. To determine which of the bond types are hydrogen, specify the keyword **h-bonds** and the number of types of hydrogen bonds, and then list the bond numbers (from the bonds section) which are considered as hydrogen bonds together with the consisting bond particle that is considered as a hydrogen atom. An example of the h-bonds section:

```
h-bonds 4
30 H+h        31 H+h        33 H2h        34 H2h
```

Hydrogen bonds are handled slightly differently than other bonds. In particular, bond angles are not formed during their creation.

2.2.6 Setting bonds updating. If the particle constituting the valent bond has changed, the Program automatically changes the bond type. To do this, it chooses the bond type is set for the changed particle(s). However, if several such types are defined, the program takes the last one from the bonds section. The **evol_bonds** section can be used to change this behavior. Specify the keyword evol_bonds and the number of bonds for which you want to define a new bond type on change. Next, indicate through - (minus sign) the initial bond type and the new bond type, for example:

```
evol_bonds 2
35-39  29-36
```

This record means that if the particles included in a bond of the 35th type have changed their type, the Program will try to change the bond type to 39. Important! The program can change the bond type in this way only if the new bond type matches the new particle types, otherwise the Program operates in a standard way, see above.

2.2.7 Setting of the bond formation. In order for the particles to spontaneously form a bond with each other, declare the linkage section. Write the keyword **linkage** and the number of pairs that can form a bond. Then, line by line, indicate the pair of particles, the minimum binding distance and the type of bond formed. Particle types and bond types must be defined in the spec and bonds sections, respectively:

```
bonds 1
1       V3      Od      harm   50.0    1.62    con     con


linkage 1
V5      O2-     1.7     1
```

In this example, particles V5 and O2-, being at a distance of 1.7 Å and less, bind to form a bond of the 1st type (section bonds). This leads to conversion of the particles into V3 and Od types, respectively, as indicated in the bond declaration. Make sure the order of the particle types in the bonds and linkage sections matches.

2.2.8 Determination of valent angles. To define the valent angle potentials, specify the **angles** keyword and the number of valent angle types. Next, write down the properties of each potential of the valent angle line by line: the serial number, the type of the central particle, the keyword of the pair potential (at the moment only hcos is supported - the harmonic cosine), the parameters of

the potential (for the harmonic cosine: the spring constant in eV and the cosine of the equilibrium angle). An example of the angles section:

```
angles 1
1       Oc      hcos    5.0     -0.33326
```

2.2.9 Definition of automatically generated valent angles. The program can automatically create valent angles if a particle of a given type is formed and it is bound by valent bonds with more than one particle. To do this, write the **angle_forming** keyword and the number of particle types for which links are automatically created. Further in each line indicates the type of particle and the type of potential of the bond angle. Particle types and valent angle potentials must be defined in the specs and angles sections:

```
angle_forming 1
Oc      1
```

2.2.10 Using a pre-created list of valent bonds and angles. For the Program to use the valent bond and angle list from the bonds.txt and angles.txt files, respectively, use the bond_list and angle_list directives with the value of 1, respectively:

```
bond_list 1
angle_list 1
```

2.3 The control.txt file

It is intended for entering general directives such as temperature, statistics output, etc. The order in which the directives are entered does not matter. Table 3 shows a complete list of directives (symbols *N* and *f* mean an integer or fractional number, respectively; different options are given through /, in brackets - an optional parameter).

Table 3. List of directives in the control.txt file

directive	**meaning**	**comment**
timestep *f*	set the step of integration equal to *f* picoseconds	Typically, 10^{-3} – 10^{-4}.
nstep *N*	set the number of the MD steps (cycles) equal to *N*	
nequil *N*	set the number of equilibration steps equal to *N*	
eqfreq *N*	scale the temperature every N steps during the relaxation period (nequil directive)	

temperature f_1 radi/nose f_2	set the temperature to f_1 (in Kelvin) radi - use a radiative thermostat nose - use a Nose-Hoover thermostat with relaxation constant of f_2 picoseconds	
init_vel zero/gaus/const f_1 f_2 f_3/keng f	set the initial velocity distribution: zero - set to zero; gaus - according to the Maxwell distribution; const f_1 f_2 f_3 - all particles have one velocity vector (f_1, f_2, f_3); keng - the modulus of velocity for all particles corresponds to the kinetic energy of f eV, the direction is random.	
reset_vels N	zero particle velocities every N steps	
permittivity f	set the dielectric constant to f, is used to calculate the Coulomb interaction.	
elec none/dir f/pme f_1 f_2 N_1 N_2 N_3/fenn f_1 f_2	set the method for the electrostatics calculation: none - no electrostatics, particle charges are ignored; dir f — pair approximation with cutoff radius of f Å; pme f_1 f_2 N_1 N_2 N_3 - use Ewald summation, f_1 - cutoff radius in real space, f_2 – the α parameter, N_1, N_2 and N_3 - number of k-vectors along the Ox, Oy and Oz axes, respectively; fenn is the Fennel and Geselter method [Fenn], f_1 is the cutoff radius, f_2 is the α parameter.	
cell_list f	set the cell size in the cell_list algorithm to f Å	
rdf f_1 f_2 N_1 N_2 (nucl)	Directive for the RDF calculation: f_1 - cutoff radius (Å), f_2 - step (Å), collect statistics every N_1 steps, output intermediate results every N_2 steps. The nucl keyword indicates to also collect RDF by summarizing atoms across nuclei (see 2.2.1)	
eJump N f_1 min/metr/eq f_2	perform the electron transfer procedure every Nth step with cutoff of f_1 Å according to:	

	min - energy minimization; metr - Metropolis scheme; eq - the condition of energy equality (Franck-Condon principle) with a permissible deviation of f_2 eV.	
Ux *f*	set the gradient of the external electric field along the x axis equal to *f* Volts / Å	
stat *N*	Output statistics files every *N*th step	

In addition, you can specify the output of the final distribution of coordination numbers (CN) for the given pairs of particle nuclei. To do this, write the **ncn** keyword and specify the number of pairs of interest. Next, list the pairs of nuclei line by line (several types of particles can have one nucleus, see 2.2.1) for which you want to derive the distribution, together with the cutoff radius for each pair. Other words in the control.txt file are ignored by the Program and can be used as comments. Example of the control.txt file:

```
timestep 0.001
nstep 500000
nequil  0
eqfreq 100

temperature 423.0    nose    0.02
init_vel        gaus

cell_list       4.0
permittivity  1.0
elec    fenn    8.0     0.4     6       6       6
rdf     10.0  0.02      50      500000          nucl

eJump 1         2.99    metr
Ux              1.0

stat            500

ncn 3
```

```
V    O    2.21
V    V    2.99
P    O    1.65
```

2.4 The cuda.txt file

Designed for video card settings. Directives:

- nstep stat - the number of statistics steps stored on the video card before being output to a file;
- nstep msdstat - the number of steps of msd statistics stored on the video card before being output to a file;
- nstep bondstat - the number of bond statistics steps stored on the video card before being output to a file;
- nthread a - the number of threads in the block for the first stage of cell_list processing;
- nthread b - the number of threads in the block for the second stage of cell_list processing.

Example of the file:

```
nstep stat 50
nstep msdstat 50
nstep bondstat 50
nthread a 16
nthread b 32
```

2.5 The bonds.txt file

Contains information about the valent bonds between particles. The first line indicates the number of valent bonds, then a triple of numbers is indicated line by line: particle indices (numbered from zero) and the type of bond (in accordance with 2.2.4). An example of the bonds.txt file (the first few lines):

```
3883
1496 1497 40
2829 2828 15
3000 3346 15
51 49 15
```

2.6 The angles.txt file

Contains information about valent angles between particles. The first line indicates the number of valent angles, then four numbers are indicated line by line: the index of the central particle, the indices of the remaining particles (numbered from zero) and the type of bond angle (in accordance with 2.2.8). An example of the angles.txt file (the first few lines):

```
1444
1152 791 1154 1
3159 3158 3157 1
1187 1507 1186 1
2609 2604 2610 1
1798 1797 2222 1
```

3 Output data

As a result of the Program's operation, the following files are generated:

- revcon.xyz (mandatory) – the size of the box and the coordinates of the particles at the end of the simulation. The same format as the input file atoms.xyz.
- revbonds.txt (optional) – the list of valent bonds at the end of the simulation is written if the valence bonds are specified in the system. Same format as the input bonds.txt file.
- revangles.txt (optional) – the list of bond angles at the end of the simulation is written if the bond angles are specified in the system. Same format as angles.txt input file.
- stat.dat (mandatory) – general system statistics every Nth step (set by the stat directive in the control.txt file, see 2.3): total energy, its components, pressure, number of particles to be changed.
- msd.dat (mandatory) – statistics on the number of particles crossing the box through each of the faces in positive and negative directions (6 values for each type of particles).
- rdf.dat (mandatory) – radial distribution function for each pair of particle types, accumulated and averaged over the entire simulation time.
- rdfN.dat (optional) – the same as rdf.dat, only accumulated and averaged to the Nth step.
- rdf_n.dat (optional) – the same as rdf.dat, only for each pair of particle nuclei. For example, if you have several types of oxygen atoms, but they have a common nucleus, they will be considered as one type (see 2.2.1).
- rdf_nN.dat (optional) – the same as rdf_n.dat, only accumulated and averaged to the Nth step.
- stat_bnd.dat (optional) – statistics on valent bonds, their number, averaged length and lifetime.

- velocities.dat (mandatory) – the list of particles' velocities at the end of the simulation. For each type of particle, the velocity modulus and its x-, y- and z-components are indicated.
- jumps.dat (optional) – statistics on electron jumps: total number, their number in the positive and negative directions along the Ox axis.

References

[Staff] A.J. Stafford, M. Silbert, J. Trullas, A. Giro, Potentials and correlation functions for the copper halide and silver iodide melts: I. Static correlations. J. Phys.: Condens. Matter. 2, 6631-6641 (1990**)** https://doi.org/10.1088/0953-8984/2/31/016

[Fenn] C. J. Fennell, J. D. Gezelter. Is the Ewald summation still necessary? Pairwise alternatives to the accepted standard for long-range electrostatics. J. Chem. Phys. 124 (2006) 234104. https://doi.org/10.1063/1.2206581