Supporting Information for

Facilitating Students' Interaction with Real Gas Properties Using a Discovery-Based Approach and Molecular Dynamics Simulations

Chelsea Sweet†, Oyewumi Akinfenwa†, Jonathan J. Foley, IV*

Department of Chemistry, William Paterson University, 300 Pompton Road, Wayne, NJ, 07470, USA

Contents of this Supporting Information Document

I. Brief theoretical background on Molecular Dynamics Simulation

A variety of computer algorithms are available that permit Newton's Law to be solved for many thousands of atoms even with modest computational resources. In this work, we utilize the Velocity Verlet algorithm to update the positions and velocities of the particles once the forces have been computed [1]. Using the Velocity Verlet algorithm, the position of the i^{th} particle is updated from time t to time $t + dt$ according to

$$\vec{r}_i(t + dt) = \vec{r}_i(t) + \vec{v}_i(t)dt + \frac{1}{2}\vec{a}_i(t)dt^2 \quad (S1)$$

where $\vec{r}_i(t)$ represents the position vector of the i^{th} particle at time t, $\vec{v}_i(t)$ represents the velocity vector of the i^{th} particle at time t, and, $\vec{a}_i(t)$ represents the acceleration vector of the i^{th} particle at time t. In our MD program, we use a Cartesian coordinate system so $\vec{r}$ has an $x, y,$ and z component: $\vec{r} = r_x\,\hat{\boldsymbol{x}} + r_y\,\hat{\boldsymbol{y}} + r_z\,\hat{\boldsymbol{z}}$ and similarly for $\vec{v}$ and $\vec{a}$. In the previous expression, $\hat{\boldsymbol{x}}$ is the unit vector along the x-axis. The velocity of the i^{th} particle is updated is updated from time t to time $t + dt$ according to

$$\vec{v}_i(t + dt) = \vec{v}_i(t) + \frac{1}{2}(\vec{a}_i(t) + \vec{a}_i(t + dt))dt \quad (S2)$$

and the acceleration comes from Newton's 2nd law, $\vec{a}_i = \frac{\vec{F}_i}{m_i}$. Note that in Eq. (S2), $\vec{a}_i(t)$ indicates the accelerations *before* the particle positions are updated via Eq. (S1) and $\vec{a}_i(t + dt)$ indicates the accelerations *after* the particle positions are updated via Eq. (S1). Hence, the velocity Verlet algorithm requires two force calculations and two calculations of the accelerations for each iteration in the simulation. An outline of the velocity Velret algorithm can be found in the block diagram that illustrates the flow of the MD simulation (Figure S1).

Practical application of the Velocity Verlet algorithm requires the specification of an initial position and initial velocity for each particle. In this work, the initial positions are

specified by arranging the atoms in a simple cubic lattice, and the initial velocities are assigned based on the Maxwell-Boltzmann distribution corresponding to a user-supplied initial temperature.

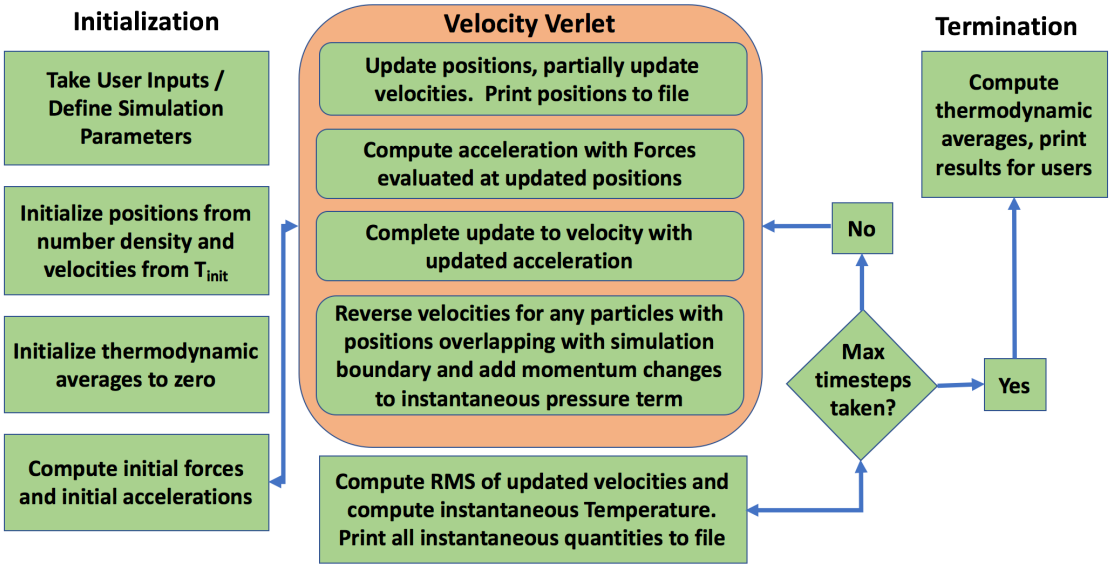


Figure S1 Block diagram illustrating flow of a Molecular Dynamics Simulation

A key ingredient in a molecular dynamics simulation is the potential energy function, also called the force field. This is the mathematical model that describes the potential energy of the simulated system as a function of the coordinates of its atoms. The potential energy function should be capable of faithfully describing the attractive and repulsive interparticle interactions relevant to the system of interest. For simulations of complex molecular or biomolecular systems with many different atom types, the potential energy function may be quite complicated, and its determination may itself be a subject of intense research (see, for example, Refs. [2-4] and references therein). Because we are interested in simulating monatomic gasses, we use a particularly simple potential energy function in our simulation known as the Lennard-Jones potential, which was also used in Rahman's original work [5],

$$U(r_{ij}) = 4\epsilon\left(\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right). \qquad (S3)$$

The Lennard-Jones potential is defined by two parameters, ϵ and σ, which may be determined by fitting simulations to experimental data, through *ab initio* calculations, or by a combination of experiment and calculation. The parameter ϵ is related to the strength of the interparticle interactions and has dimensions of energy. The value of ϵ manifests itself as the minimum value of the potential energy between a pair of particles, $\min\left(U(r_{ij})\right) = -\epsilon$, as shown in the plot of the Lennard-Jones potential in Figure S2. The parameter σ is related to the effective size of the particles and has dimensions of length. In effect, the value of σ determines at which separations attractive forces dominate and at which separations repulsions dominate. For example, the value of σ determines the interparticle separation that minimizes the Lennard-Jones potential by the relation $r_{min} = 2^{\frac{1}{6}}\sigma$ where $U(r_{min}) = -\epsilon$, as shown in Figure S2. Importantly, only the scalar separation between a pair of particles is needed to determine the potential energy of the pair within the Lennard-Jones model. The total potential energy of the

system of N particles is simply the sum over the potential energy of all unique pairs. This potential neglects orientation effects which may be important for describing molecules that lack spherical symmetry, and also neglects polarization effects which may arise in many chemically relevant situation, water for example [6]. Lennard-Jones parameters for various noble gasses are provided in Table 1.

Table S1 Lennard-Jones parameters for several noble gasses along with particle mass. Lennard-Jones parameters for He, Ne, Kr, Xe are taken from Ref. [7], and parameters for Ar are taken from Ref. [8].

Particle	m **(kg)**	σ **(m)**	ϵ **(J)**
Helium	6.646e-27	2.64e-10	1.5e-22
Neon	3.350e-26	2.74e-10	5.6e-22
Argon	6.633e-26	3.3605e-10	1.9618e-21
Krypton	1.391e-26	3.58e-10	2.75e-21
Xenon	2.180e-26	3.80e-10	3.87e-21

Once the potential energy function has been specified, the forces on the particles may be calculated from the derivative on the potential energy function with respect to interparticle separation *vectors:*

$$\vec{F}_i = -\sum_{j \neq i}^{N} \frac{\partial U(\vec{r}_{ij})}{\partial \vec{r}_{ij}} . \qquad (S4)$$

That is, each particle experiences a unique force from each of the remaining particles in the system. Each unique force is related to the derivative of the potential energy with respect to the separation vector of the particle pair,

$$\vec{F}_{ij}(\vec{r}_{ij}) = \frac{24\epsilon}{\sigma^2}\, \vec{r}_{ij} \left(2 \left(\frac{\sigma}{\vec{r}_{ij}} \right)^{14} - \left(\frac{\sigma}{\vec{r}_{ij}} \right)^{8} \right) \qquad (S5).$$

Note that unlike the potential energy, the pair force is fundamentally a vector quantity and has a direction associated with it. We note that the two $\frac{\sigma}{\vec{r}_{ij}}$ terms in the force can be equivalently evaluated in terms of the scalar separation, r_{ij}, because they contain even powers of the separation vector, and even powers of the separation vector are equivalent to the same even power of the scalar separation. Because the Lennard-Jones potential has a minimum at the separation $r_{min} = 2^{\frac{1}{6}}\sigma$, the force goes to zero at separation r_{min} (*see Figure S*1). Once all the unique pair force vectors $\vec{F}_{ij}$ are determined from evaluation of Equation (S5) based on the coordinates of the system, the total force vector $\vec{F}_i$ acting on the i^{th} particle is computed as the sum over all the unique pair forces. Hence, the potential energy function is a key ingredient in determining the dynamics through its impact on the forces through Equation (S4), and therefore, the instantaneous position and velocity through Equations (S1) and (S2), respectively. The instantaneous positions and velocities of the particles constitutes the raw data generated by a molecular dynamics simulation. These trajectories may be directly visualized using animation software (e.g. VMD, Ref [9]), and a variety of physical quantities can be derived from information contained in these trajectories.

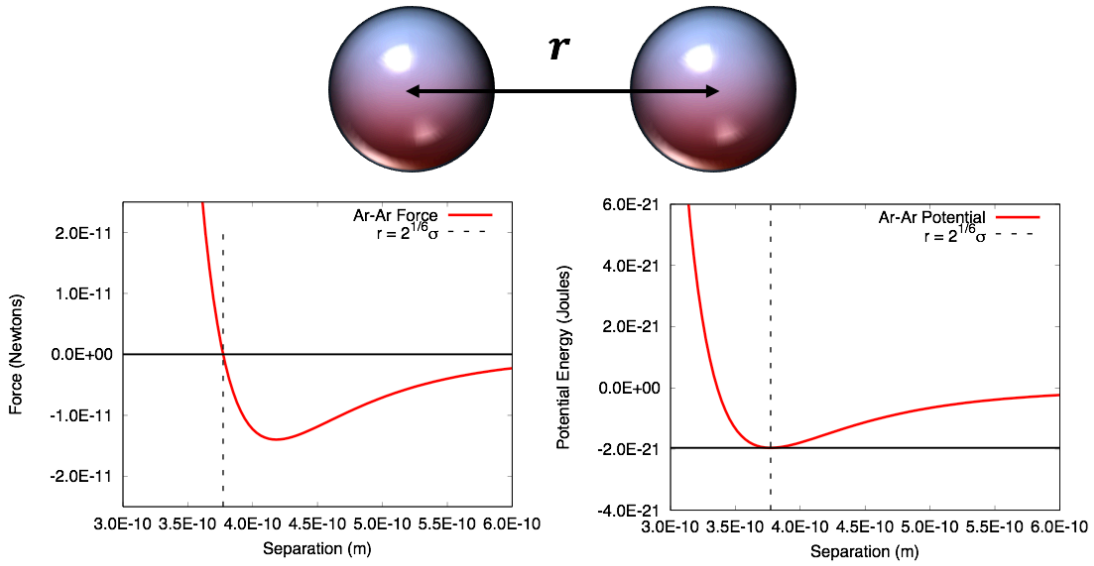
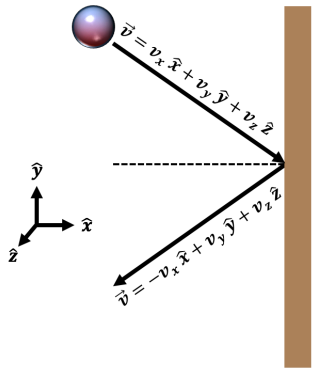


Figure S2 Illustration of Lennard-Jones force (left plot) and Lennard-Jones potential (right plot) as a function of separation between two argon atoms. All quantities are plotted in SI units where $\epsilon = 1.961 \cdot 10^{-21}\ J$ and $\sigma = 3.605 \cdot 10^{-10}\ m$ for Argon. Note that the potential energy is a minimum, and the force is zero, at the separation $r = 2^{\frac{1}{6}}\sigma$. A complete table of L-J parameters in SI units for the noble gasses is provided in Table S1.

Two relevant macroscopic quantities that can be derived from molecular dynamics trajectories are the pressure and temperature of the system. As in kinetic theory, the pressure in our program is taken to arise from the changes in particle momentum associated with elastic collisions with their "container walls", i.e., the boundaries that constrain the particles being simulated. We take the gas system to be constrained by a cubic box with a user-supplied volume. All gas particles in the simulation are given initial positions within the boundaries of this box, and if a gas particle's position reaches the boundary along a particular axis (for example, the *x*-axis), the corresponding (*x-)* component of velocity is simply reversed, simulating an elastic collision (see Figure S3). An instantaneous force exerted on the wall can be defined by accounting for all such momentum changes that occur in a particular time interval,

$$F_{wall}(t) = \sum_{i}^{N} \frac{\Delta\, \vec{p}_i(t)}{\Delta t} = \sum_{k} \sum_{\xi} \frac{2m\, v_{k,\xi}(t)}{\Delta t}. \qquad (S6)$$

In this expression, the index k runs only over the particles that suffer collisions with the walls, and the index ξ runs over only the components of the velocity that change upon collision with the walls. Figure S3 illustrates a particle colliding elastically with a wall and experiencing a change in the *x*-component of its velocity vector; this collision results in a change in the *x*-component of the particles momentum vector with magnitude $\Delta p_x = 2m\, v_x$.



Figures S3 *Schematic of an elastic collision of a particle with its container wall. The particle reaches the container boundary along the x-axis, and so experiences a change in its x-component of velocity. This change in velocity is associated with a change in the x-component of momentum, as well. The sum over all such momentum changes in a given time interval can be interpreted as the instantaneous pressure within the kinetic theory of gasses.*

Similarly, within kinetic theory, the instantaneous temperature can be related to the mean squared velocity at a given instant in time by

$$T(t) = \frac{m\langle v^2(t)\rangle}{3k_B} \qquad (S7)$$

where the mean squared velocity is defined as

$$\langle v^2(t)\rangle = \frac{1}{N}\sum_{i}^{N}\sum_{\xi} \vec{v}^2{}_{i,\xi}(t), \qquad (S8)$$

m is the particle mass and k_B is Boltzmann's constant, and the sum over ξ sums over the $x, y,$ and z components of velocity. The thermodynamic pressure and temperature can be approximated time average of the instantaneous $T(t)$ and $P(t)$ over the duration of the simulation; for Temperature, this average is computed as

$$T = \frac{1}{N_{steps}}\sum_{i=1}^{N_{steps}} T(t_i), \qquad (S9)$$

and the Pressure average is computed analogously.

For readers interested in inspecting the source code that implements these equations, we note that our underlying Molecular Dynamics code relies on a natural unit system in which the particle mass (m), the Lennard-Jones interaction strength (ϵ) and size parameter (σ), and Boltzmann's constant k_B are all equal to 1. Each noble gas therefore has a unique set of conversion factors for relating quantites in natural units to quantities in SI units. We provide a selection of such conversion factors in Table S2 for each noble gas as a companion to the Lennard-Jones and atomic mass data in SI units provided in Table S1.

Table S2 Factors for converting several quantities from natural units to SI units. For example, one natural unit of volume for Helium is equal to 1.84e-29 L^3.

Particle	**Volume (σ^3)**	**Temperature (ϵ/k_B)**	**Pressure (ϵ/σ^3)**	**Time $\left(\sqrt{\frac{m\sigma^2}{\epsilon}}\right)$**
Helium	$1.84\cdot 10^{-29}$	10.	$8.1\cdot 10^{6}$	$1.7\cdot 10^{-12}$
Neon	$2.06\cdot 10^{-29}$	40.	$2.7\cdot 10^{7}$	$2.1\cdot 10^{-12}$
Argon	$4.685\cdot 10^{-29}$	142.095	$4.1874\cdot 10^{7}$	$2.09618\cdot 10^{-12}$
Krypton	$4.58\cdot 10^{-29}$	199.	$5.99\cdot 10^{7}$	$8.05\cdot 10^{-12}$
Xenon	$5.48\cdot 10^{-29}$	280.	$7.05\cdot 10^{7}$	$9.01\cdot 10^{-13}$

II. Programming Exercise

Developing computer code the computes the temperature and pressure from molecular dynamics trajectories can provide students a unique opportunity to interact with kinetic theory equations. The act of translating the equations into computer instructions requires students to break down the mathematics into elementary steps, which can aid their comprehension of the underlying equations. Furthermore, the process will require students to manipulate the microscopic quantities that give rise to the trajectories they see rendered as animations; this connection can also help students build conceptual bridges between the microscopic forces that lead to gas behavior and the macroscopically observable properties of the gas. Further suggestions for executing this programming exercise can be found in the Supporting Information.

The following steps provide further information to aid the programming exercises. In particular, we highlight the parts of the code that perform the desired functions. Instructors who wish to provide these programming challenges may delete parts or all of the code shown below before introducing the program to their students, and work with the students to build the desired functionality back in by planning and writing similar code themselves.

This exercise involves computing thermodynamic quantities (temperature and pressure) from the molecular dynamics trajectories by using kinetic theory expressions. Note that the trajectory information (position, velocity, and acceleration), as well as the forces, are stored as 2D arrays r[][], v[][], a[][], F[][], respectively. These arrays have global scope so their data can be accessed and manipulated to all functions in the program. In all of these arrays, the inner-index references the particle number and the outer-index references the Cartesian component. For example, the x-position of particle 1 is stored in r[0][0] and the z-component of velocity of particle 99 is stored in v[98][2] (array indices start from 0 in c).

As discussed in the main text, instantaneous contributions to the pressure and temperature can be computed from trajectories using Kinetic Theory. In particular, the instantaneous pressure relies on summing up the momentum changes associated with wall collisions. The part of the code that updates the positions and velocities of the particles at each time-step must account for wall collisions anyway, so this is a natural place to compute these contributions. These updates occur in the function called 'VelocityVerlet', and the pressure contributions are accumulated in the variable 'psum' in the following code block:

~/MolecularDynamics

```
double VelocityVerlet(double dt, int iter, FILE *fp) {
  int i, j, k;

  double psum = 0.;

  //  Compute accelerations from forces at current position
  computeAccelerations();
  //  Update positions and velocity with current velocity and acceleration
  //printf("  Updated Positions!\n");
  for (i=0; i<N; i++) {
    for (j=0; j<3; j++) {
      r[i][j] += v[i][j]*dt + 0.5*a[i][j]*dt*dt;

      v[i][j] += 0.5*a[i][j]*dt;
    }
    //printf("  %i  %6.4e   %6.4e   %6.4e\n",i,r[i][0],r[i][1],r[i][2]);
  }
  //  Update accellerations from updated positions
  computeAccelerations();
  //  Update velocity with updated acceleration
  for (i=0; i<N; i++) {
    for (j=0; j<3; j++) {
      v[i][j] += 0.5*a[i][j]*dt;
    }
  }

  // Elastic walls
  for (i=0; i<N; i++) {
    for (j=0; j<3; j++) {
       if (r[i][j]<0.) {
         v[i][j] *=-1.; //- elastic walls
         psum += 2*m*fabs(v[i][j])/dt;  // contribution to pressure from "left" walls
       }
       if (r[i][j]>=L) {
         v[i][j]*=-1.;  //- elastic walls
         psum += 2*m*fabs(v[i][j])/dt;  // contribution to pressure from "right" walls
       }
    }
  }
```

The instantaneous contribution to the temperature depends upon the instantaneous mean squared velocity, which is computed in the following block of code in the function called 'MeanSquaredVelocity':

~/MolecularDynamics

```
//  Function to calculate the averaged squared velocity
double MeanSquaredVelocity() {

  double vx2 = 0;
  double vy2 = 0;
  double vz2 = 0;
  double v2;

  for (int i=0; i<N; i++) {

    vx2 = vx2 + v[i][0]*v[i][0];
    vy2 = vy2 + v[i][1]*v[i][1];
    vz2 = vz2 + v[i][2]*v[i][2];

  }
  v2 = (vx2+vy2+vz2)/N;


  return v2;
}
-- INSERT --                                                    258,56        52%
```

To estimate the thermodynamic temperature and pressure of the simulated system, we take the average of the instantaneous quantities over each time step. Hence, a sum of instantaneous temperatures and pressures is accumulated over each time-step in the simulation, and then divided by the total number of time steps at the end:

~/MolecularDynamics

```
for (i=0; i<NumTime; i++) {

  // This updates the positions and velocities using Newton's Laws
  // Also computes the Pressure as the sum of momentum changes from wall collisions / timestep
  // which is a Kinetic Theory of gasses concept of Pressure
  Press = VelocityVerlet(dt, i+1, tfp);
  Press *= PressFac;
  //  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
  //  Now we would like to calculate somethings about the system:
  //  Instantaneous mean velocity squared, Temperature, Pressure
  //  Potential, and Kinetic Energy
  //  We would also like to use the IGL to try to see if we can extract the gas constant
  mvs = MeanSquaredVelocity();
  KE = Kinetic();
  PE = Potential();

  // Temperature from Kinetic Theory
  Temp = m*mvs/(3*kB) * TempFac;

  // Instantaneous gas constant and compressibility - not well defined because
  // pressure may be zero in some instances because there will be zero wall collisions,
  // pressure may be very high in some instances because there will be a number of collisions
  gc = NA*Press*(Vol*VolFac)/(N*Temp);
  Z  = Press*(Vol*VolFac)/(N*kBSI*Temp);

  Tavg += Temp;
  Pavg += Press;

  fprintf(ofp,"  %8.4f  %20.8f  %20.8f %20.8f  %20.8f  %20.8f \n",i*dt,Temp,Press,KE, PE, KE+PE);


}

// Because we have calculated the instantaneous temperature and pressure,
// we can take the average over the whole simulation here
Pavg /= NumTime;
Tavg /= NumTime;
                                                          194,11        33%
```

~/MolecularDynamics

```
// Because we have calculated the instantaneous temperature and pressure,
// we can take the average over the whole simulation here
Pavg /= NumTime;
Tavg /= NumTime;
Z = Pavg*(Vol*VolFac)/(N*kBSI*Tavg);
gc = NA*Pavg*(Vol*VolFac)/(N*Tavg);
                                                          191,20        37%
```

REFERENCES

1. Verlet, L. Computer “Experiments” on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Phys. Rev.* **1967,** *159*, 98.
2. Karplus, M.; McCammon, J. A. Molecular dynamics simulations of biomolecules. *Nat. Struct. Bio.* **2002,** *9*, 646-652.
3. Brooks, B. R.; Brooks III, C. L.; Mackerell Jr., A. D; Nilsson, L.; Petrella, R. J.; Roux, B.; Won, Y.; Archontis, G.; Bartels, C.; Boresch, S.; Caflisch, A.; Caves, L.; Cuis, Q.; Dinner, A. R.; Feig, M.; Fischer, S.; Gao, J.; Hodoscek, M.; Im, W.; Kuczera, K.; Lazaridis, T.; Ma, J.; Ovchinnikov, V.; Paci, E.; Pastor, R. W.; Post, C. B.; Pu, J. Z.; Schaefer, M.; Tidor, B.; Venable, R. M.; Woodcock, H. L.; Wu, X.; Yang, W.; York, D. M.; Karplus, M. CHARMM: The biomolecular simulation program. *J. Comp. Chem.* **2009,** *30*, 1545-1614.

4. Senftle, T. P.; Hong, S.; Islam, M. M.; Kylasa, S. B.; Zheng, Y.; Shin, Y. K.; Junkermeier, C.; Engel-Herbert, R.; Janik, M. J.; Aktulga, H. M.; Verstaelen, T.; Grama, A.; van Duin, A. C. T. The ReaxFF reactive force-field: development, applications, and future directions *Comp. Mater.* **2016**, 2, 15011
5. Rahman, A. Correlations in the Motion of Atoms in Liquid Argon. *Phys. Rev.* **1964,** *136*, A405.
6. Sillinger, F. H.; Rahman, A. Improved simulation of liquid water by molecular dynamics. *J. Chem. Phys.* **1974,** *60*, 1545.
7. Schroeder, D. V. Interactive molecular dynamics. *Am. J. Phys.* **2015,** *83*, 210-218.
8. Barker, J. A.; Fisher, R. A.; Watts, R. O. Liquid argon: Monte carlo and molecular dynamics calcualtions. *Mol. Phys.* **1971**, *21*, 657-673
9. Humphrey, W.; Dalke, A.; Schulten, K. VMD: Visual Molecular Dynamics. *J. Mol. Graph.* **1996,** *14*, 33-38.