Symulacja kryształu argonu — sprawozdanie

Bartłomiej Baur

31 października 2022

1 Wstęp

Program *Argon* symuluje kryształ argonowy. Uruchomienie programu wymaga trzech argumentów:

./argon <plik_CONFIG> <plik_OUT> <plik_XYZ>

gdzie <plik_CONFIG> to plik konfiguracyjny o przykładowej strukturze przedstawionej w sekcji 4.2, <plik_OUT> to plik, w którym zapiszą się wyniki wyjściowe symulacji (energia, temperatura, ciśnienie), natomiast <plik_XYZ> to plik, w którym zostaną zapisane położenia atomów w kolejnych krokach czasowych.

2 Symulowanie kryształu i analiza parametrów makroskopowych układu.

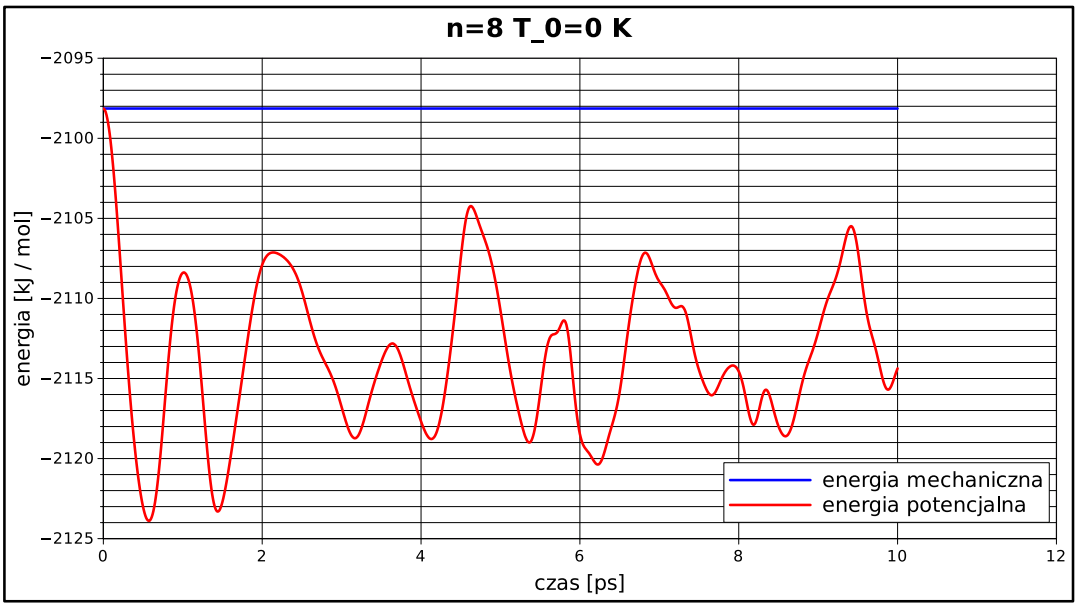
Za pomocą napisanego programu przeprowadzono szereg symulacji dla kryształów dla $n \in \{6, 7, 8\}$ oraz $T_0 \in \{0, 1000\}$. W pliku wyjściowym OUT zapisują się energie całkowita, potencjalna i kinetyczna oraz chwilowa temperatura, i chwilowe ciśnienie układu. W poniższych podsekcja przedstawiono wyniki przykładowych z przeprowadzonych symulacji.

2.1 n=8, T_0=0 K

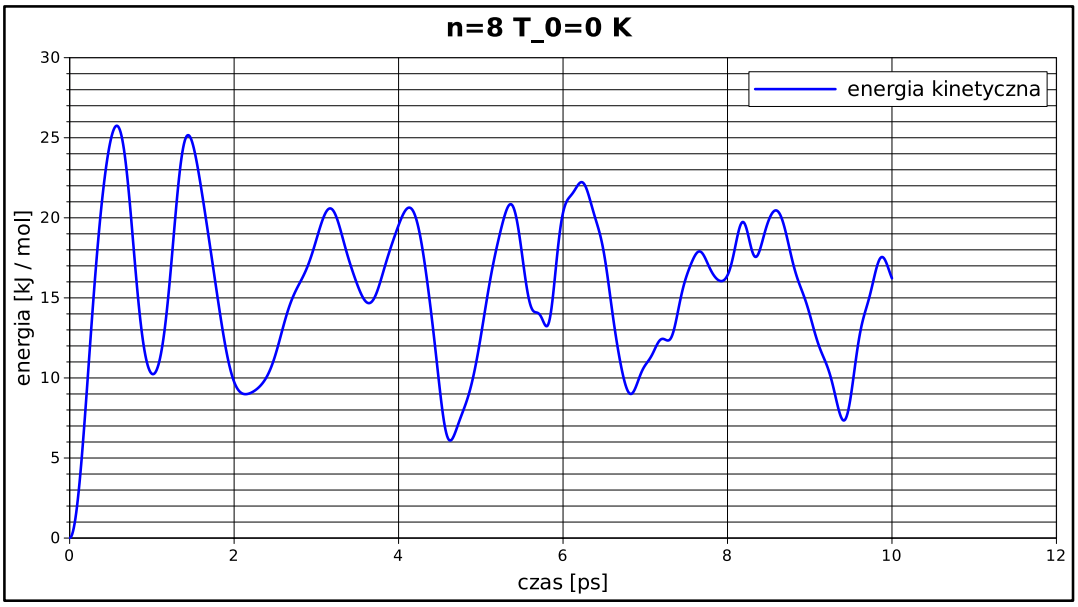
Krawędź kryształu ma tutaj 8 atomów, co daje nam $8^3 = 512$ atomów w symulacji. Ustawiono tutaj temperaturę zera bezwzględnego. Jakakolwiek energia kinetyczna a więc i chwilowa temperatura wynika jedynie z sił oddziaływań między atomami i dążeniem kryształu do minimum potencjału. Wykresy dla energii zostały przedstawione na grafikach 1 oraz 2.

Na wykresie 3 przedstawiono wykres zależności chwilowej temperatury od czasu. Widać, że wykres ten ma ten sam kształt, co wykres energii kinetycznej, co wynika z liniowej zależności między tymi wielkościami.

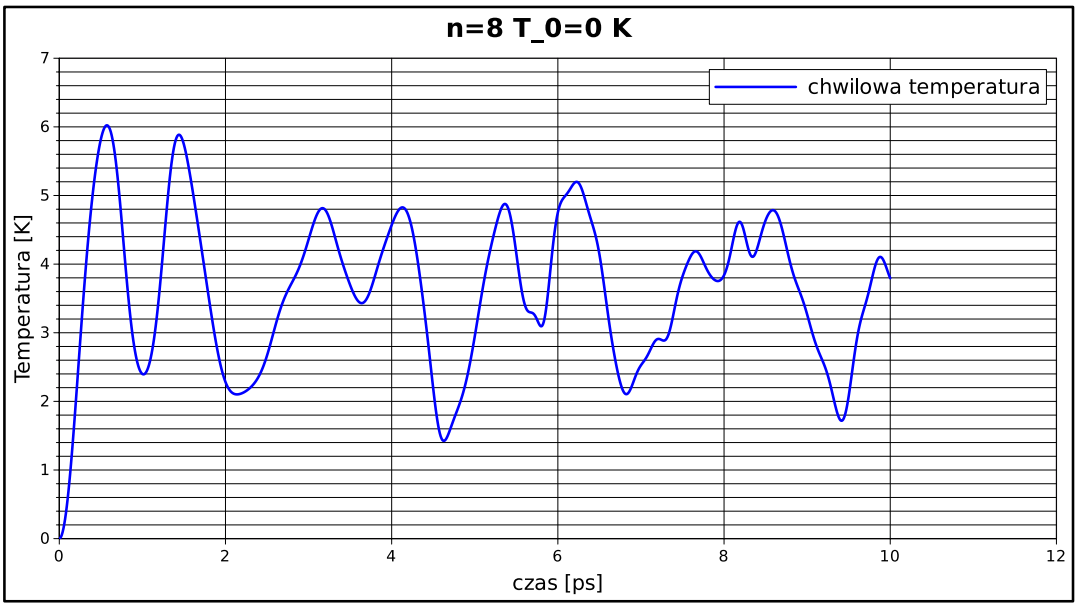
Wykres ciśnienia na Grafice 4 utrzymuje wartość zero. Staje się to oczywiste, gdy zwróci się uwagę na fakt, że solidny kryształ nie oddziałuje z przygotowanymi przez nas „ściankami naczynia"



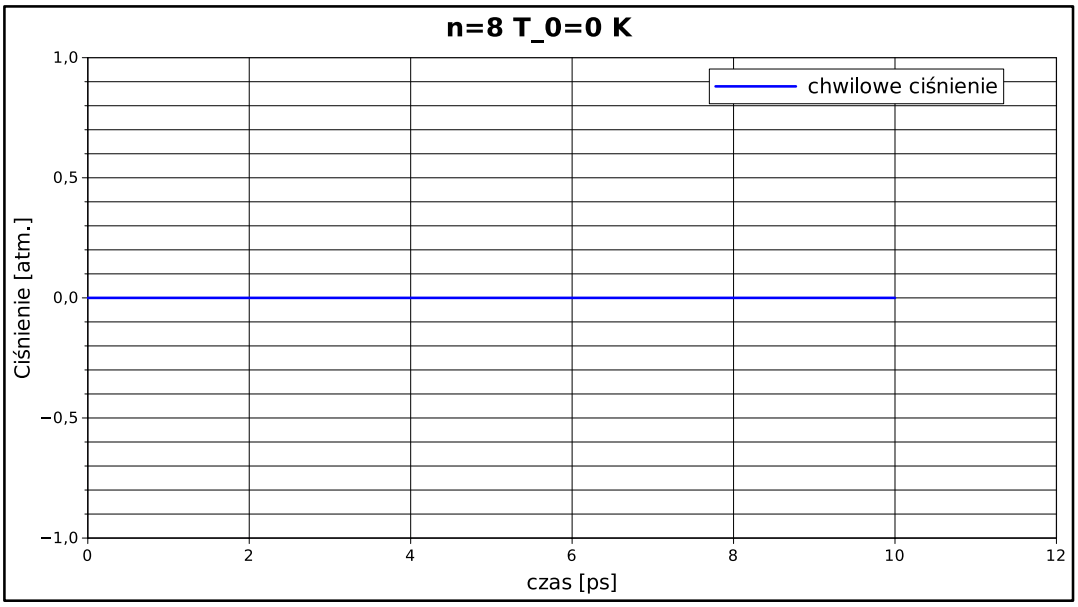
Grafika 1: Zależność całkowitej energii mechanicznej oraz energii potencjalnego od czasu symulacji.



Grafika 2: Zależność energii kinetycznej od czasu w symulacji dla temperatury początkowej 0 K.



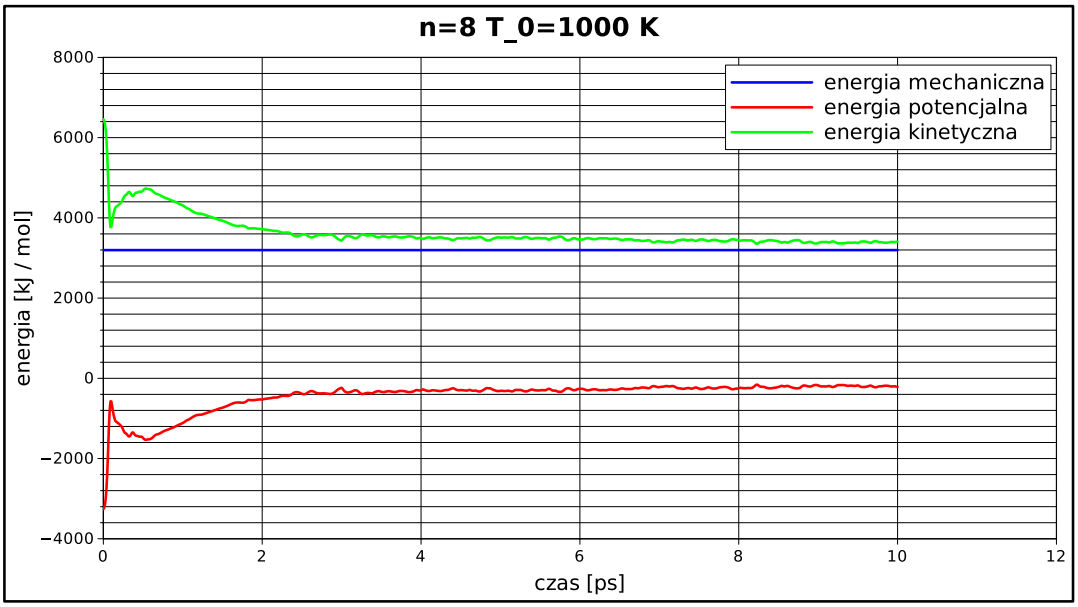
Grafika 3: Wykres chwilowej temperatury od czasu w symulacji dla temperatury początkowej 0 K.



Grafika 4: Wykres ciśnienia od czasu w symulacji dla temperatury początkowej 0 K.

2.2 n=8, T_0=1000 K

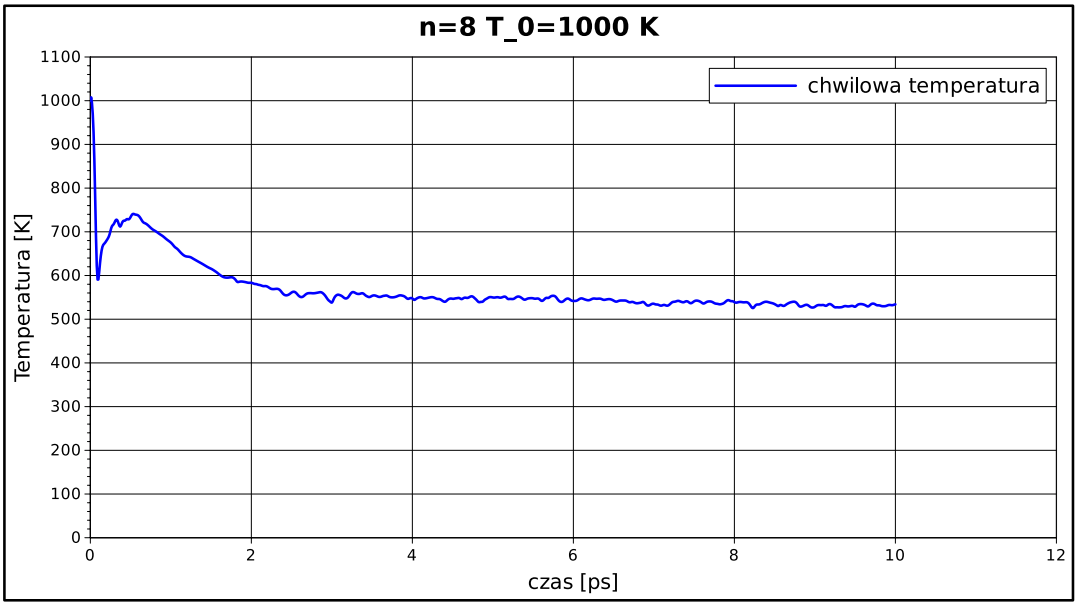
Dla temperatury początkowej 1000 K kryształ rozpada się i argon przyjmuje gazowy stan skupienia. Zmianę widać na wykresie energii układu przedstawionym na Grafice 5, gdzie widać, że potencjał układu jest bliski zera, energia kinetyczna zaś stanowi większość energii mechanicznej.



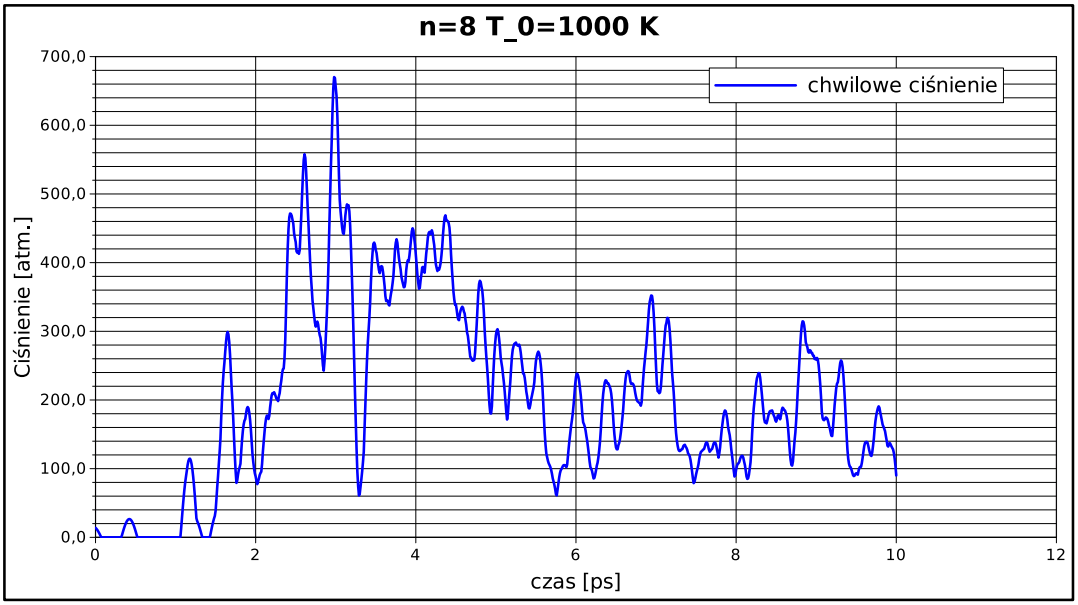
Grafika 5: Zależność energii mechanicznej, kinetycznej i potencjalnej od czasu w symulacji dla temperatury początkowej 1000 K.

Wykres chwilowej temperatury przedstawiony na wykresie 6 ponownie ma kształt odpowiadający wykresowi energii kinetycznej. Widać, że średnia temperatura układu będzie wyższa, niż poprzednio.

Na wykresie z Grafiki 7 widać, że ciśnienie ulega silnym fluktuacjom. Wynika to z tego, że ciśnienie jest tylko wtedy, gdy cząstki wpadają w obszar potencjału tworzącego „ścianki naczynia", w którym trzymamy nasz wirtualny kryształ.



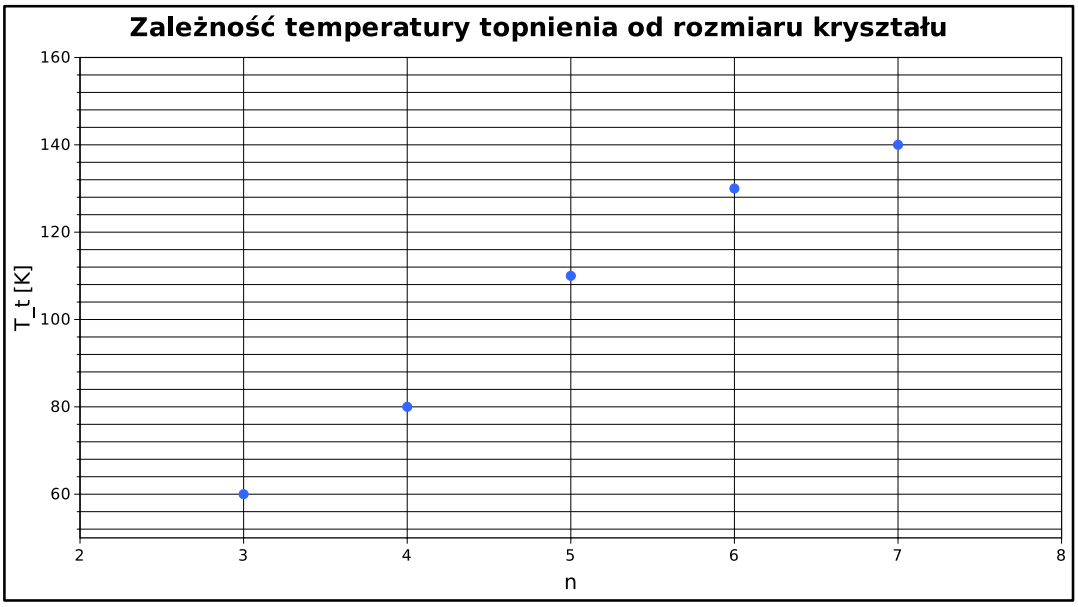
Grafika 6: Zależność chwilowej temperatury od czasu w symulacji dla temperatury początkowej 1000 K.



Grafika 7: Wykres ciśnienia od czasu w symulacji dla temperatury początkowej 0 K.

3 Badanie temperatury topnienia kryształu

Drugim etapem prowadzonych badań z użyciem symulacji było określenie temperatury topnienia kryształu argonu. W przeprowadzonych symulacjach próbowano zgadnąć temperaturę topnienia dla kryształów o różnych rozmiarach. Okazuje się, że symulacja nie jest doskonała, bo temperatura ta zależy od ilości atomów w krysztale, co jest sprzeczne z rzeczywistością. Patrząc na wykres przedstawiony na Grafice 8 można zauważyć, że zależność ta jest w przybliżeniu liniowa.



Grafika 8: Wykres zależności temperatury topnienia symulowanego kryształu od rozmiaru kryształu. n oznacza liczbę atomów w krawędzi kryształu.

4 Źródło programu

Program do symulacji kryształu został napisany w języku C++ i wykorzystuje bibliotekę *Simple Config File* autorstwa Fernando Barber Miralles[1], aby móc wykonywać ładne pliki konfiguracyjne.

4.1 Kod programu

```
#include <iostream>
#include <cmath>
#include <vector>
#include <fstream>
#include <cstdlib>
#include <ctime>
#include "SCF/config_file.h"

//parametry programu
int n, N, S, S_out, S_xyz;
double m, e, R, f, L, a, T0, tau;
const double epsilon = 1; // kJ/mol
const double k = 8.31e-3; // kJ/(K*mol)


//parametry symulacji
std::vector<double> r, p, F, Eki;
double V=0, Ek=0, E;
double T, P;
double VP=0, VS=0, FP=0, FS=0;

//zmienne pomocnicze
double dr;
double tmp1, tmp2, tmp3;

void load_parameters(const char* param_file_) {
  std::ifstream param;
  param.open(param_file_);
  std::vector<std::string> ln = {"n", "m", "e", "R", "f", "a", "T_0", "tau", "S", "
    S_out", "S_xyz"};
  CFG::ReadFile(param, ln, n, m, e, R, f, a, T0, tau, S, S_out, S_xyz);
  param.close();
  N = n*n*n;
  L = 1.22 * a * n;
  std::cout<<"Zaladowano parametry z pliku "<<param_file_<<std::endl;
}

void r_init() {
  //inicjalizacja polozen
  const std::vector<double> b0{ a, 0, 0 };
  const std::vector<double> b1{ a/2, a*sqrt(3)/2, 0 };
  const std::vector<double> b2{ a/2, a*sqrt(3)/6, a*sqrt(2.0/3) };

  for(int i2=0; i2<n; i2++)
  for(int i1=0; i1<n; i1++)
  for(int i0=0; i0<n; i0++)
    for(int j=0; j<3; j++) {
```

```
47        double tmp = (i0-(n-1)/2)*b0[j] + (i1-(n-1)/2)*b1[j] + (i2-(n-1)/2)*b2[j];
48        r.push_back(tmp);
49      }
50  }
51
52  void p_init() {
53    //losujemy Ek oraz znak pedu, nastepnie liczymy ped
54    for(int i=0; i<3*N; i++) {
55      double lambda = (double) (std::rand()%10000)/10000;
56      double E = -0.5 * k * T0 * log(lambda);
57      int signum = (std::rand()%2) % 2 == 0 ? 1 : -1;
58      p.push_back(signum * sqrt(2*m*E));
59    }
60    //normujemy ped tak, aby dla calego ukladu wynosil zero
61    double P[3] = {0,0,0};
62    for(int i=0; i<N; i++) {
63      for(int j=0; j<3; j++) {
64        P[j]+=p[3*i+j];
65      }
66    }
67    P[0]/=N;
68    P[1]/=N;
69    P[2]/=N;
70    for(int i=0; i<N; i++) {
71      for(int j=0; j<3; j++) {
72        p.at(3*i+j)-=P[j];
73      }
74    }
75  }
76
77  void calc_EkandE() {
78
79    //energia kinetyczna
80    Ek=0;
81    for(int i=0; i<N; i++) {
82      for(int j=0; j<3; j++) {
83        Eki[i] = p[3*i+j]*p[3*i+j];
84        Ek += Eki[i];
85      }
86    }
87    Ek /= 2*m;
88
89    //energia calkowita
90    E = Ek + V;
91  }
92
93  void calc_VFP() {
94
95    P = 0;
96    V = 0;
97    for(int i=0; i<3*N; i++)
98      F[i] = 0;
99
100   for(int i=0; i<N; i++) {
101
```

```
102     dr = sqrt( r[3*i]*r[3*i] + r[3*i+1]*r[3*i+1] + r[3*i+2]*r[3*i+2] ); //odleglosc
        od srodka ukladu odniesienia
103     tmp1 = dr<L ? 0 : 0.5*f*(dr-L)*(dr-L); //potencjal od scianek VS
104     V += tmp1;
105     for(int j=0; j<3; j++) { //sila od scianek FS
106             F[3*i+j] += dr<L ? 0 : f*(L-dr)*r[3*i+j]/dr;
107     }
108     P += sqrt( F[3*i]*F[3*i] + F[3*i+1]*F[3*i+1] + F[3*i+2]*F[3*i+2] ); //
       akumulacja cisnienia. Bedzie ono jeszcze mnozone przez wspolczynnik!
109     for(int j=0; j<i; j++) {
110       dr = sqrt( (r[3*i]-r[3*j])*(r[3*i]-r[3*j]) + (r[3*i+1]-r[3*j+1])*(r[3*i+1]-r
       [3*j+1]) + (r[3*i+2]-r[3*j+2])*(r[3*i+2]-r[3*j+2]) ); //od teraz dr to odleglosc
        miedzy dwoma atomami!!!
111       tmp1 = R/dr; //rachunek pomocniczy
112       tmp2 = tmp1 * tmp1 * tmp1 * tmp1 * tmp1 * tmp1; //rachunek pomocniczy
113       tmp3 = epsilon * (  tmp2*tmp2 - 2*tmp2  ); //potencjal miedzy atomami VP
114       V += tmp3;
115       for(int k=0; k<3; k++) { //sily van der Waalsa FP
116         tmp3 = 12 * epsilon * (tmp2*tmp2 - tmp2) * (r[3*i+k]-r[3*j+k]) / dr / dr;
       //k-ta skladowa sily pomiedzy i-tym i j-tym atomem
117         F[3*i+k] += tmp3;
118               F[3*j+k] += -tmp3;
119       }
120     }
121   }
122   P /= 4*M_PI*L*L; //konczenie rachunkow P
123
124 }
125
126
127 int main(int argc, const char *argv[]) {
128
129   //inicjalizacja programu
130   if(argc!=4) {
131     std::cout<<"Uzycie:"<<std::endl;
132     std::cout<<"argon <parametry> <wyjscie energii> <wyjscie wspolrzednych>"<<std::
       endl;
133     return -1;
134   }
135   srand( time( NULL ) );
136   load_parameters(argv[1]);
137
138   std::ofstream xyz, out;
139   xyz.open(argv[3]);
140   out.open(argv[2]);
141   std::cout<<"Otworzono plik "<<argv[2]<<" do zapisu danych wyjsciowych programu."
       <<std::endl;
142   std::cout<<"Otworzono plik "<<argv[3]<<" do zapisu wspolrzednych symulowanych
       atomow."<<std::endl;
143
144   out<<"t\t\tE\t\tEk\t\tV\t\tT\t\tP"<<std::endl;
145   std::cout<<"OK"<<std::endl;
146
147   //inicjalizacja symulacji
148   std::cout<<"Inicjalizacja symulacji... ";
149   r_init();
```

```
150   p_init();
151   F.reserve(3*N);
152   Eki.reserve(N);
153   calc_VFP();
154   calc_EkandE();
155
156   //wypisanie poczatkowych polozen
157   xyz<<N<<std::endl<<std::endl;
158   for(int i=0; i<N; i++) {
159     xyz<<"Ar ";
160     for(int j=0; j<3; j++) {
161       xyz<<r[3*i+j]<<" ";
162     }
163     xyz<<std::endl;
164   }
165   //zapisanie sytuacji poczatkowej do OUT.txt
166   out<<0<<"\t\t"<<E<<"\t\t"<<Ek<<"\t\t"<<V<<"\t\t"<<T0<<"\t\t"<<P<<std::endl;
167
168   std::cout<<"OK"<<std::endl;
169
170
171
172   std::cout<<"Rozpoczeto symulacje..."<<std::endl;
173   //petla symulacji
174   for(int step=1; step<=S; step++) {
175
176     for(int i=0; i<3*N; i++) {
177       p[i] = p[i] + 0.5*F[i]*tau;
178       r[i] = r[i] + p[i]*tau/m;
179     }
180     calc_VFP();
181     for(int i=0; i<3*N; i++) {
182           p[i] = p[i] + 0.5*F[i]*tau;
183     }
184     calc_EkandE();
185     T = 2*Ek/3/N/k;
186
187
188     if(step%S_out==0) {
189       out<<tau*step<<"\t\t"<<E<<"\t\t"<<Ek<<"\t\t"<<V<<"\t\t"<<T<<"\t\t"<<P<<std::
    endl;
190     }
191     if(step%S_xyz==0) {
192       xyz<<N<<std::endl<<std::endl;
193       for(int i=0; i<N; i++) {
194         xyz<<"Ar"<<" ";
195         for(int j=0; j<3; j++) {
196           xyz<<r[3*i+j]<<" ";
197         }
198         xyz<<std::endl;
199       }
200     }
201   }
202   std::cout<<"Obliczenia zakonczone!"<<std::endl;
203
204   xyz.close();
```

```
205   out.close();
206   return 0;
207 }
208
```

4.2 Plik konfiguracyjny

Treść przykładowego pliku konfiguracyjnego:

```
n=5
m=40
e=1
R=0.38
f=1e4
a=0.38
T_0=0
tau=1e-3
S=10000
S_out=10
S_xyz=100
```

Literatura

[1] F. B. Miralles biblioteka „Simple Config File”
`https://github.com/fbarberm/SimpleConfigFile`