Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчет по предмету:
Параллельная обработка данных в научных исследованиях

Де Ен Де, 508

Москва, 2023

Содержание

1. Постановка задачи

Требуется определить потенциалы межатомного взаимодействия A-A, A-B и B-B для системы A/B(001). Виды потенциалов:

$$E = \sum_i \left(E_R^i + E_B^i\right)$$

$$E_R^i = \sum_j \left(A_{\alpha\beta}^1 \left(r_{ij} - r_0^{\alpha\beta}\right) + A_{\alpha\beta}^0\right) \exp\left(-p_{\alpha\beta}\left(\frac{r_{ij}}{r_0^{\alpha\beta}} - 1\right)\right)$$

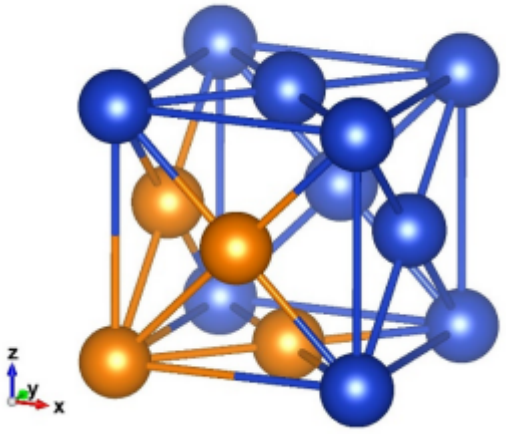
$$E_B^i = -\left(\sum_j \xi_{\alpha\beta}^2 \exp\left(-2q_{\alpha\beta}\left(\frac{r_{ij}}{r_0^{\alpha\beta}} - 1\right)\right)\right)^{\frac{1}{2}}$$

- E – полная энергия системы;
- E_R^i и E_B^i – энергии отталкивания и притяжения i-ого атома
- r_{ij} – расстояние между i-ым и j-ым атомами;
- $A_{\alpha\beta}^0, A_{\alpha\beta}^1, r_0^{\alpha\beta}, p_{\alpha\beta}, q_{\alpha\beta}, \xi_{\alpha\beta}$ – параметры потенциалов

Расчеты полной энергии проводятся для кристаллической решетки размером 3x3x3 в единицах элементарной ячейки ГЦК. Все расчеты проводятся без релаксации атомных позиций. Атомы располагаются в узлах идеальной ГЦК решетки

2. Вычисление характеристик

Гранецентрированная кубическая решетка(ГЦК) представлена на рис. 1а). Для построения решетки нужно выделить элементарную ячейку, состоящей из 4 базисных атомов (0,0,0), (0.5a, 0.5a, 0), (0.5a, 0, 0.5a) и (0, 0.5a, 0.5a) (атомы выделены желтым цветом). Далее необходимо размножить элементарную ячейку до решетки 3x3x3, как указано на рисунке 1.



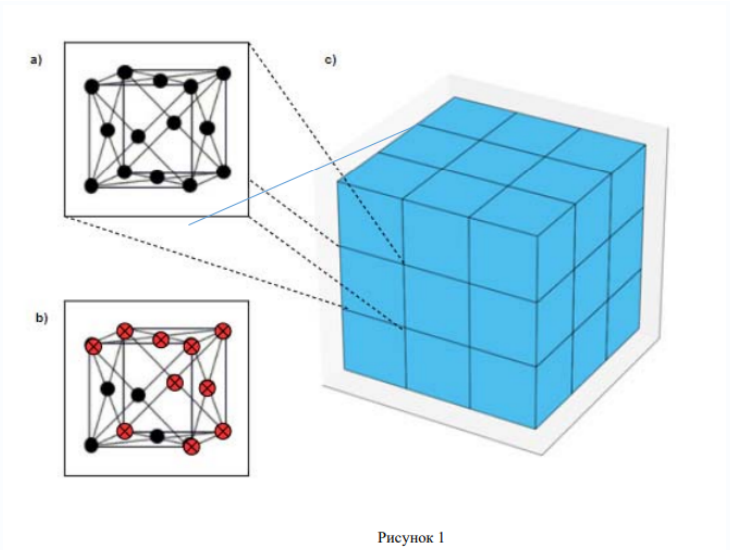


Рисунок 1

Для решения задачи необходимо найти набор параметров модельного потенциала, который позволит получить заданные характеристики материала. Построим функцию оптимизации как взвешенную сумму квадратов отклонений характеристик.

Функционал имеет следующий вид:

$$F(params) = \sum_f W_f \frac{(f(params) - f^*)^2}{f^{*2}},$$

$$f \in (a, Ecoh, B, C11, C12, C44, Esol, E_{\text{dim}}^{in}, E_{\text{dim}}^{on})$$

, где

- f (params) – вычисленные значения;
- f*– эталонные значения.

Рассмотрим характеристики материала:

1. a – параметр решетки, который находится как точка минимума полной энергии системы *E* по 5 точкам из возможного диапазона значений;
2. *Ecoh* – когезионная энергия, которая вычисляется по следующей формуле:

$$E_{coh} = \frac{E}{N}$$

- *E* – полная энергия;
- *N* – количество атомов в решетке;

3. B – модуль всестороннего растяжения/сжатия:

$$B = \frac{2}{9V_0} \frac{\partial^2 E_B}{\partial \alpha^2}$$

- *V*0 – равновесный объем;

Для вычисления этого параметра была использована матрица деформации, которая имеет следующий вид:

$$D = \begin{pmatrix} 1+\alpha & 0 & 0 \\ 0 & 1+\alpha & 0 \\ 0 & 0 & 1+\alpha \end{pmatrix}$$

4. C_{11}, C_{12}, C_{44} – константы упругости:

$$C_{11} = \frac{1}{V_0}\frac{\left(\frac{\partial^2 E_{C_{11}}}{\partial \alpha^2} + \frac{\partial^2 E_{C_{12}}}{\partial \alpha^2}\right)}{2}$$

$$C_{12} = \frac{1}{V_0}\frac{\left(\frac{\partial^2 E_{C_{11}}}{\partial \alpha^2} - \frac{\partial^2 E_{C_{12}}}{\partial \alpha^2}\right)}{2}$$

$$C_{44} = \frac{1}{V_0}\frac{\partial^2 E_{C_{44}}}{\partial \alpha^2}$$

Для вычисления этих параметров были использованы матрицы деформации, которые имеют следующий вид:

$$D11 = \begin{pmatrix} 1+\alpha & 0 & 0 \\ 0 & 1+\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$D12 = \begin{pmatrix} 1+\alpha & 0 & 0 \\ 0 & 1-\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$D44 = \begin{pmatrix} 1 & \alpha & 0 \\ \alpha & 1 & 0 \\ 0 & 0 & \frac{1}{1-\alpha^2} \end{pmatrix}$$

5. E_{sol} – энергия растворимости примеси рода A в кристалле рода B:

$$E_{sol} = E^{AB} - E^{B} - E^{A}_{coh} - E^{B}_{coh}$$

- E_{AB} – полная энергия кристаллической решетки B с примесью замещения A;
- E_B – полная энергия кристаллической решетки B;
- E^{A}_{coh} – когезионная энергия A (берется из справочника);
- E^{B}_{coh} – когезионная энергия B;

6. E^{dim}_{in} – энергия связи димера A в поверхностном слое B:

$$E_{dimin} = (E_{dim+surf} - E_{surf}) * 2(E_{adatom+surf} - E_{surf})$$

- $E_{dim+surf}$ – полная энергия структуры поверхности с димером A в верхнем слое;
- $E_{adatom+surf}$ – полная энергия структуры поверхности с одним атомом A в верхнем слое;
- E_{surf} – полная энергия структуры поверхности;

7. E^{dim}_{on} – энергия связи димера A на поверхности B:

$$E^{on}_{dim} = \left(E^{dim+surf} - E^{surf}\right) * 2\left(E^{adatom+surf} - E^{surf}\right)$$

- $E_{dim+surf}$ – полная энергия структуры поверхности с димером A над верхним слоем;
- $E_{adatom+surf}$ – полная энергия структуры поверхности с одним атомом A над верхним слоем;
- E_{surf} – полная энергия структуры поверхности;

3. Метод оптимизации

Для решения задачи оптимизации в работе используется метод Нелдера-Мида. Рассмотрим его алгоритм.

В основе этого метода лежит движении симплекса в пространстве параметров в сторону локального минимума целевой функции.

Параметрами метода являются:

- Коэффициент отражения $\alpha > 0$, обычно выбирается равным 1.
- Коэффициент сжатия $\beta > 0$, обычно выбирается равным 0.5.
- Коэффициент растяжения $\gamma > 0$, обычно выбирается равным 2.
- Коэффициент глобального сжатия $\sigma > 0$, обычно выбирается равным 0.5.

1. Инициализация метода. Случайным образом выбираются $nn + 1$ точка в nn - мерном пространстве параметров, образующие симплекс, и в них вычисляются значения целевого функционала:

$$x_i = (x_i^{(1)}, x_i^{(2)}, \dots, x^{(n)i}), F(x_i) = f_i \ .$$

2. . Выберем из вершин симплекса три: x_h , x_g и x_l , такие что f_h — наибольшее значение целевой функции из всех вершин симплекса, f_g — второе по величине значение и f_l — наименьшее.
3. Найдём центр тяжести всех точек за исключением x_h :

$$x_c = \frac{1}{n}\sum_{i \neq h} x_i \ .$$

4. Отразим точку x_h относительно точки x_c с коэффициентом α. Получим таким образом точку $x_r = (1 + \alpha)x_c - \alpha x_h$ и вычислим в ней значение целевого функционала f_r.

5. Проверим насколько нам удалось улучшить значение функции:

- $f_r < f_l$ — направление выбрано удачно, попробуем увеличить шаг. Вычислим новую точку $x_e = (1 - \gamma)x_c + \gamma x_r$ и значение функции в ней f_e. Из точек x_r и x_e выбираем наилучшую и заменяем ею x_h.
- $f_l \le f_r < f_g$ — новая точка улучшает ответ, заменим ею x_h.
- $f_g \le f_r < f_h$ — новая точка улучшает ответ, но слабо, заменим ею x_h и проведём операцию сжатия.
- $f_h \le f_r$ — новая точка не улучшает ответ, проведём операцию сжатия.

6. Сжатие. Если было решено провести операцию сжатия, то построим новую точку $x_s = (1 - \beta)x_c + \beta x_h$ и вычислим значение функции в ней f_s.
- $f_s < f_h$ — заменяем вершину x_h точкой x_s.
- $f_h \le f_s$ — сжимаем весь симплекс к точке с наименьшим значением $x_i \leftarrow x_l + \sigma(x_i - x_l)$.

4. Описание программы

Программа на вход ничего не принимает. Конфигурационные данные берутся из файла Const.hpp.

Выходными данными программы является вывод, в котором указаны полученные параметры потенциалов для трех взаимодействий (A-A, A-B, B-B).

Пример вывода программы:

B-B: A0 = 0.1028; A1 = 1.17054e-310; ksi = 1.178; p = 10.928; q = 3.139; a0 = 4.085

A-B: A0 = 0.169988; A1 = 0.233594; ksi = 1.41237; p = 11.2608; q = 3.27181; a0 = 3.68578

A-A: A0 = 0.0457012; A1 = 0.00583804; ksi = 1.72314; p = 10.8348; q = 3.38821; a0 = 4.50424

Также программа выводит значений целевой функции и параметры для данной ошибки.

Error: 0.668221A0 = 0.17155; A1 = 0.234375; ksi = 1.4155; p = 11.2593; q = 3.27025; a0 = 3.68812

Программа состоит из 12 файлов:

Dot.cpp, EnergyFun.cpp, Optimizer.cpp, Parameters.cpp, main.cpp, Const.hpp, DeformParams.hpp, Dot.hpp, EnergyFun.hpp, Optimizer.hpp, Parameters.hpp, Params.hpp

Также для сборки присутствует программы Makefile

Основные классы и структуры:

```
//Dot - координаты атома и признак того, является ли атом примесью
struct Dot {
    double x, y, z;
    bool isDim = false;
    Dot& operator-= (const Dot& o);
    Dot& operator*= (double o);
    Dot friend operator-(const Dot& lhs, const Dot& rhs);
    Dot friend operator*(const Dot& lhs, double rhs);
    Dot friend operator*(double lhs, const Dot& rhs);
    friend bool operator==(const Dot &lhs, const Dot &right);
    friend bool operator==(const Dot *lhs, const Dot &right);
    friend std::ostream  &operator<<(std::ostream &out, const Dot &d);
```

```
};


// DeformParams - структура хранящая модуль всестороннего растяжения и константы упругости
struct DeformParams {
    double B;
    double C11;
    double C12;
    double C44;
};

// Parameters - класс для работы с параметрами, которые будут оптимизироваться
class Parameters
{
private:
    double A0;
    double A1;
    double p;
    double ksi;
    double q;
    double a0;
public:
    double getA0() const;
    double getA1() const;
    double getP() const;
    double getKsi() const;
    double getQ() const;
    double getCubeSide() const;
    Parameters &withA0(const double &A0);
    Parameters &withA1(const double &A1);
    Parameters &withP(const double &p);
    Parameters &withKsi(const double &ksi);
    Parameters &withQ(const double &q);
    Parameters &withCubeSide(const double &a0);

    Parameters operator+(const Parameters &rhs);
    Parameters operator*(const double &dig);
    Parameters operator-(const Parameters &rhs);
    Parameters &operator+=(const Parameters &rhs);
    Parameters &operator/=(const double &dig);
    friend std::ostream &operator<<(std::ostream &out, const Parameters &param);
};
```

```
//Optimizer - класс, выполняющий задачу оптимизации методом Нелдера-Мида
class Optimizer
{
private:
    std::multimap<double, Parameters> simplex;
    Parameters BBParams;
    Parameters ABParams;
    Parameters AAParams;
    double ref_coef = 1.0;
    double compr_coef = 1.0;
    double stretch_coef = 0.5;
    double glob_compr_coef = 0.5;
    void shrink(Parameters &x_c, Parameters &x_h, Parameters &x_l, const
double &f_h, double (* error)(const Parameters &params));
public:
    static double errorBB(const Parameters &params);
    static double errorAB(const Parameters &params);
    static double errorAA(const Parameters &params);
    void initBB();
    void initAB();
    void initAA();
    void step(double (* error)(const Parameters &params));
    void run();
    const Parameters &getAAParams() const;
    const Parameters &getABParams() const;
    const Parameters &getBBParams() const;

};
```

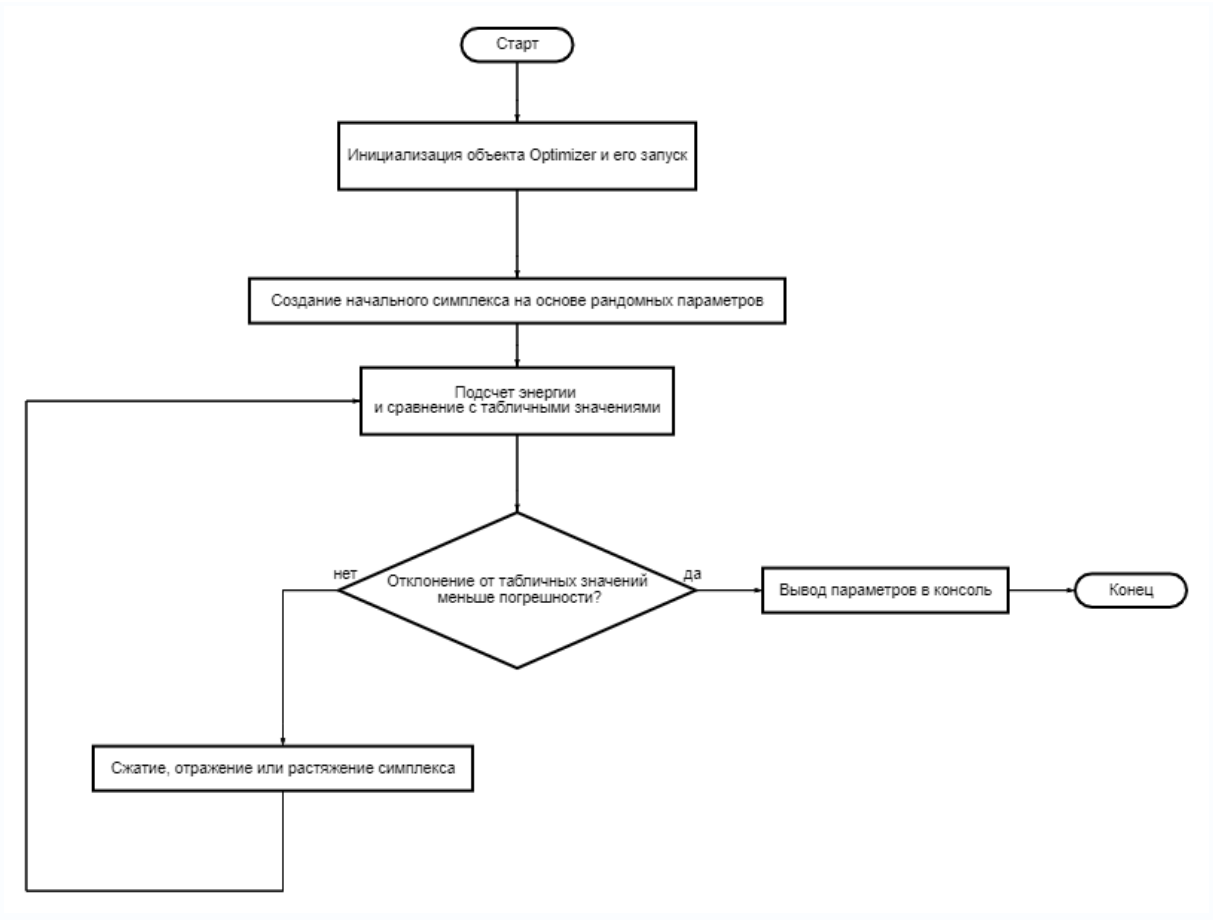


Схема работы программы

5. Распараллеливание программы

Одной из самых затратных операций является подсчет полной энергии кристаллической решетки E. За эти вычисления отвечают функции:

```
double countEnergy(const std::vector<Dot*> &data, const Parameters &params,
const double d[9]);
double countSolEnergy(const std::vector<Dot*> &data, const Parameters
&params, const double d[9]);
double countSurfEnergy(const std::vector<Dot*> &data, const Parameters
&params, const double d[9]);
```

Данные функции используют директиву

```
#pragma omp parallel for reduction
```

Применяется она для выделения отдельного потока для каждой итерации цикла, так как подсчет энергии взаимодействия между двумя атомами кристаллической решетки не зависит от остальных атомов решетки. Редукция оператором + происходит для переменной `sumE`, которая хранит в себе суммарную энергию решетки.

Код программы:

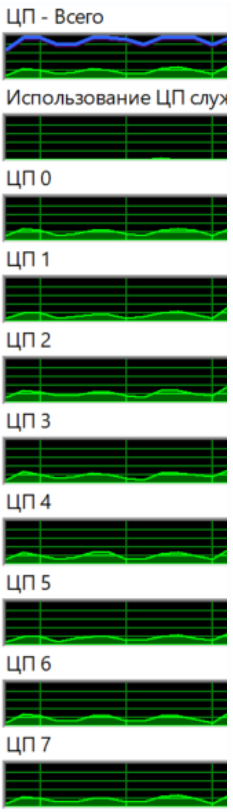
```
double countEnergy(const std::vector<Dot*> &data, const Parameters &params,
const double d[9]) {
    double sumE = 0;
    double a0 = params.getCubeSide();
    double cutOff = CONST::CUT_OFF * a0;
    double r0 = a0 / sqrt(2);

    #pragma omp parallel for reduction(+: sumE)
    for (unsigned i = 0; i < data.size(); ++i) {
        Dot dot0 = countDefformation(*data[i], d);
        double curEB = 0;
        double curER = 0;
        #pragma omp parallel for reduction(+: curEB, curER)
        for (unsigned j = 0; j < data.size(); ++j) {
            for (int dx = -1; dx < 2; ++dx) {
                for (int dy = -1; dy < 2; ++dy) {
                    for (int dz = -1; dz < 2; ++dz) {
                        if (i == j && dx == 0 && dy == 0 && dz == 0) {
                            continue;
                        }
                        Dot dotBorder = borderCondition(*data[j], dx, dy,
```

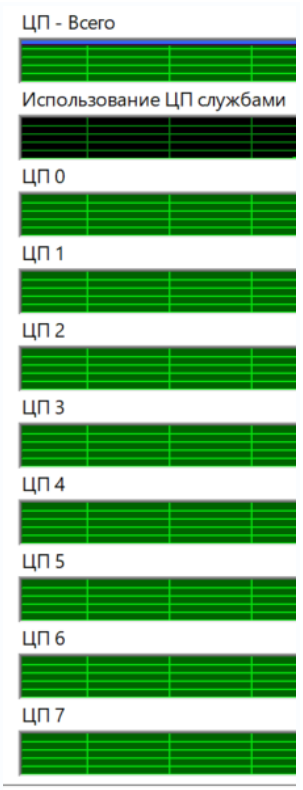
```
dz, a0);
                        Dot dot1 = countDefformation(dotBorder, d);
                        double dist = sqrt(
                            pow(dot1.x - dot0.x, 2) + pow(dot1.y - dot0.y,
2) + pow(dot1.z - dot0.z, 2)
                        );
                        if (dist > cutOff) {
                            continue;
                        }
                        curER += params.getA0() * exp(-params.getP() * (dist
/ r0 - 1));
                        curEB += pow(params.getKsi(), 2) * exp(-2 *
params.getQ() * (dist / r0 - 1));
                    }
                }
            }
        }
        sumE += (-sqrt(curEB) + curER);
    }
    return sumE;
}
```

Использование данной директивы позволило ускорить выполнение вычислений в несколько раз:

- без распараллеливания: 15.78s
- с распараллеливанием: 2.37s



Ресурс ЦП в однопоточном режиме



Ресурс ЦП в многопоточном режиме

6. Анализ полученных параметров

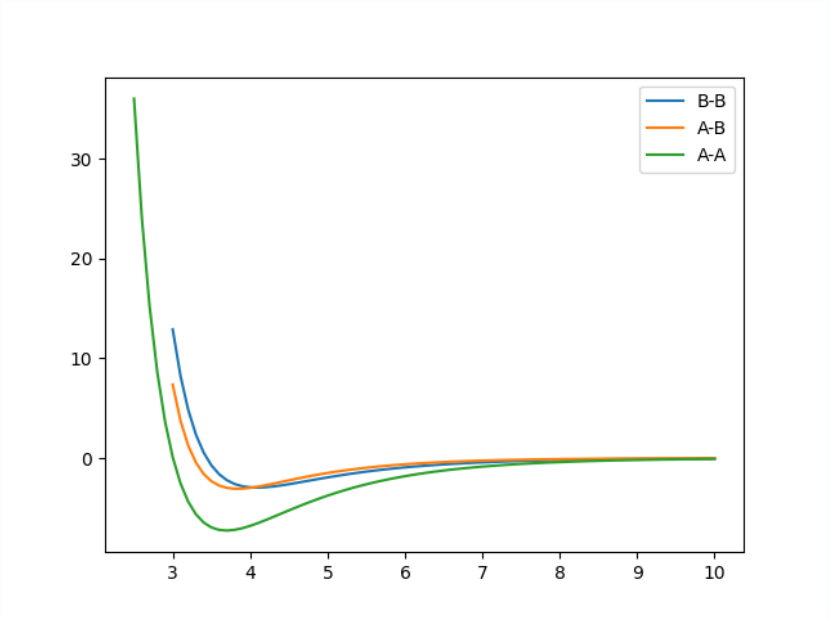
В ходе работы были получены оптимальные параметры:

B-B: A0 = 0.1028; A1 = 1.17054e-310; ksi = 1.178; p = 10.928; q = 3.139; a0 = 4.085
A-B: A0 = 0.169988; A1 = 0.233594; ksi = 1.41237; p = 11.2608; q = 3.27181; a0 = 3.68578
A-A: A0 = 0.0457012; A1 = 0.00583804; ksi = 1.72314; p = 10.8348; q = 3.38821; a0 = 4.50424

Для этих параметров были получены следующие значения потенциалов:
Ecoh = -2.9603 B = 1.08303 C11 = 1.31656 C12 = 0.966251 C44 = 0.506432
ESol = 0.632767
EInDim = -0.0794742 EOnDim = -0.678188

Все значения потенциалов примерно равны табличным. Также для проверки были получены значения энергии кристаллической решетки для каждого типа взаимодействия. Для B-B была посчитана ECoh, для A-B - ESol и для A-A была посчитана ESurf без примеси. Значения энергий были получены для a0 в диапазоне между [3.0; 10.0]

Полученные результаты были визуализированы средствами Python и Matplotlib. Полученные графики имеют правильную форму и доказывают корректность полученных параметров:



7. Заключение

В данной работе для поиска параметров потенциала использовался метод Нелдера-Мида, являющийся методом нулевого порядка. Данный алгоритм не требует сложных вычислений и потребляет много ресурсов процессора и памяти. С другой стороны данный метод обладает большим недостатком в виде отсутствия критерия сходимости, что может привести к неправильному результату. Также возможна ситуация, когда рабочий симплекс находится далеко от оптимальной точки, а алгоритм производит большое число итерации, при этом мало изменяя значения целевой функции.

Использование параллелизма дает прирост к скорости выполнения вычислений, что позволило быстро анализировать результаты программы в ходе ее проектирования и написания. При более сложной структуре решетки и большего количества элементов в ней моделирование такой системы было бы затруднительным без использования данной технологии.