Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра информационной безопасности

Мирпулатов Исломбек Пулат-угли

Многомасштабное моделирование физических явлений и процессов

ЛАБОРАТОРНАЯ РАБОТА №1

Преподаватель:

К.К. Абгарян

А.А. Журавлев

Москва, 2022

Постановка задачи

В рамках лабораторной работы нам дан материал, его группа симметрий и позиции Уайкова атомов:
Fe(2+) 2Al(3+) 4O(2-) в позициях Уайкова 8b 16c 32e соответственно.

Необходимо найти минимальную константу решётки при которой может существовать подобное соединение, позицию свободного атома и изобразить полученную кристаллическую структуру.

Решение и результаты

Ионные радиусы для атомов:

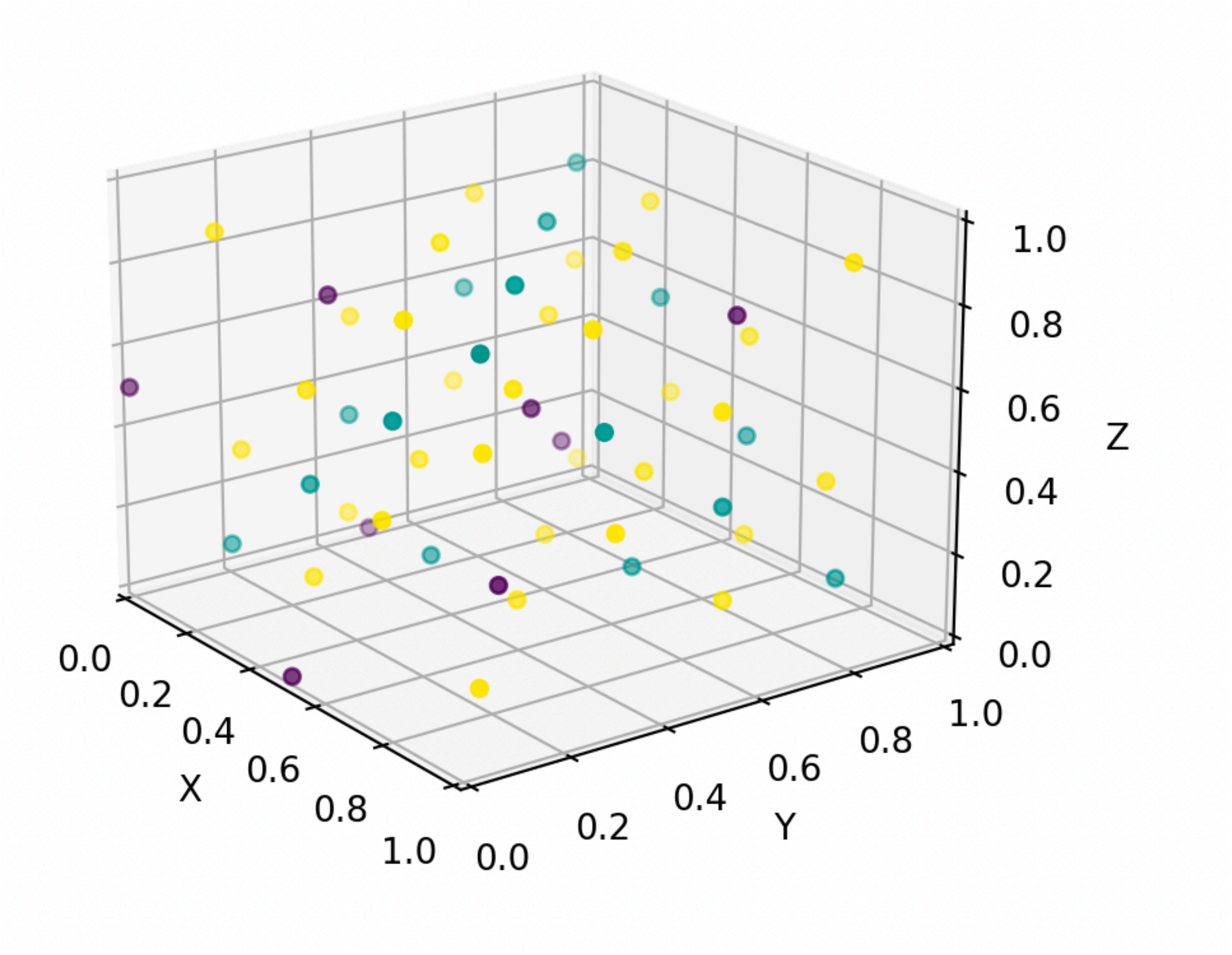
b	Fe	0.74
c	Al	0.51
e	O	1.32

В результате подсчётов получены следующие результаты:

- Параметр решётки в диапазоне от 8.4892578125 до 8.49609375
- Расположения всех 56 атомов:

```
[0.5, 0.5, 0.5], 'Fe', 0.74
[0. , 0. , 0.5], 'Fe', 0.74
[0.75, 0.75, 0.75], 'Fe', 0.74
[0. , 0.5, 0. ], 'Fe', 0.74
[0.75, 0.25, 0.25], 'Fe', 0.74
[0.5, 0. , 0. ], 'Fe', 0.74
[0.25, 0.75, 0.25], 'Fe', 0.74
[0.25, 0.25, 0.75], 'Fe', 0.74
[0.125, 0.125, 0.125], 'Al', 0.51
[0.125, 0.875, 0.875], 'Al', 0.51
[0.375, 0.375, 0.125], 'Al', 0.51
[0.875, 0.125, 0.875], 'Al', 0.51
[0.625, 0.375, 0.875], 'Al', 0.51
[0.375, 0.875, 0.625], 'Al', 0.51
[0.375, 0.125, 0.375], 'Al', 0.51
[0.875, 0.875, 0.125], 'Al', 0.51
[0.375, 0.625, 0.875], 'Al', 0.51
[0.625, 0.125, 0.625], 'Al', 0.51
```

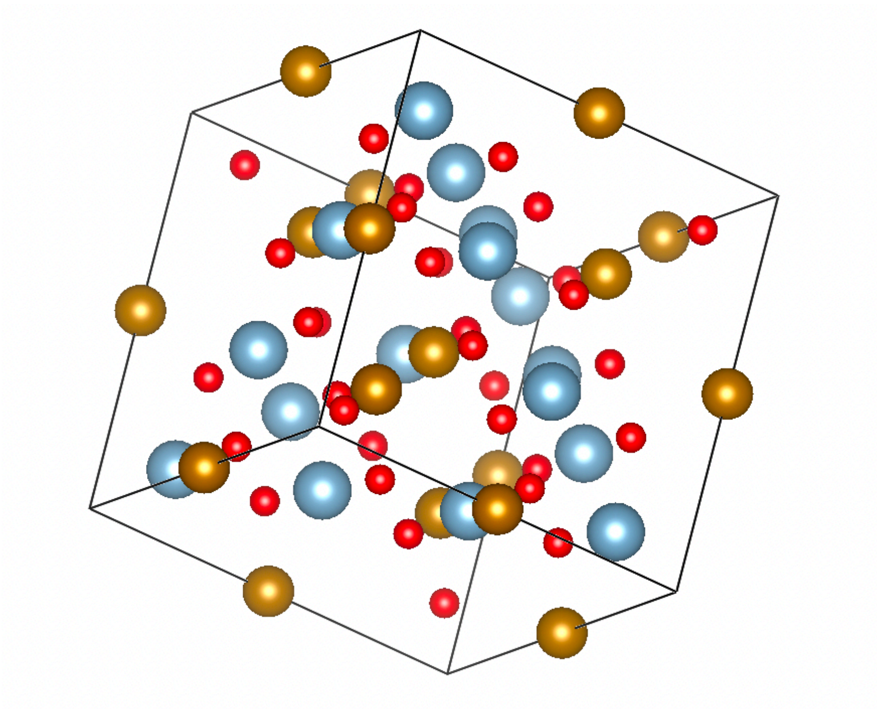
```
[0.625, 0.875, 0.375], 'Al', 0.51
[0.875, 0.375, 0.625], 'Al', 0.51
[0.125, 0.375, 0.375], 'Al', 0.51
[0.625, 0.625, 0.125], 'Al', 0.51
[0.125, 0.625, 0.625], 'Al', 0.51
[0.875, 0.625, 0.375], 'Al', 0.51
[0.36, 0.36, 0.36], 'O', 1.32
[0.36, 0.64, 0.64], 'O', 1.32
[0.14, 0.14, 0.36], 'O', 1.32
[0.89, 0.89, 0.89], 'O', 1.32
[0.64, 0.36, 0.64], 'O', 1.32
[0.86, 0.14, 0.64], 'O', 1.32
[0.61, 0.61, 0.89], 'O', 1.32
[0.14, 0.64, 0.86], 'O', 1.32
[0.14, 0.36, 0.14], 'O', 1.32
[0.89, 0.11, 0.11], 'O', 1.32
[0.64, 0.64, 0.36], 'O', 1.32
[0.14, 0.86, 0.64], 'O', 1.32
[0.61, 0.89, 0.61], 'O', 1.32
[0.86, 0.36, 0.86], 'O', 1.32
[0.86, 0.64, 0.14], 'O', 1.32
[0.61, 0.11, 0.39], 'O', 1.32
[0.64, 0.14, 0.86], 'O', 1.32
[0.39, 0.61, 0.11], 'O', 1.32
[0.36, 0.14, 0.14], 'O', 1.32
[0.11, 0.89, 0.11], 'O', 1.32
[0.86, 0.86, 0.36], 'O', 1.32
[0.89, 0.61, 0.61], 'O', 1.32
[0.61, 0.39, 0.11], 'O', 1.32
[0.36, 0.86, 0.86], 'O', 1.32
[0.39, 0.89, 0.39], 'O', 1.32
[0.64, 0.86, 0.14], 'O', 1.32
[0.11, 0.61, 0.39], 'O', 1.32
[0.39, 0.11, 0.61], 'O', 1.32
[0.11, 0.11, 0.89], 'O', 1.32
[0.89, 0.39, 0.39], 'O', 1.32
[0.39, 0.39, 0.89], 'O', 1.32
[0.11, 0.39, 0.61], 'O', 1.32
```



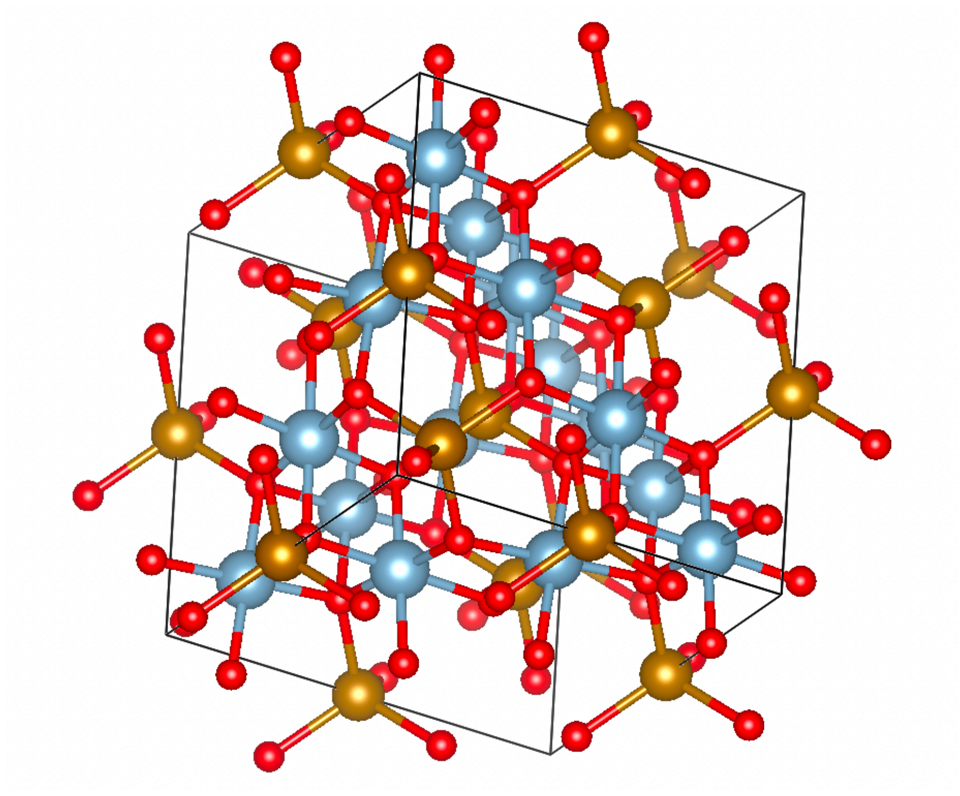
Визуализация в VESTA

Вводим полученные в прошлом этапе данные в VESTA.

Получаем структуру без связей:



Добавляем связи между элементами структуры:



```
===================================================================
Title                   FEAL2O4

Lattice type            F
Space group name        F d -3 m
Space group number      227
Setting number          1

Lattice parameters

   a         b         c        alpha     beta      gamma
 8.49260   8.49260   8.49260   90.0000   90.0000   90.0000

Unit-cell volume = 612.522541 Å^3

Structure parameters

                            x          y          z          Occ.      U
Site      Sym.
   1 Fe   Fe          0.50000    0.50000    0.50000    1.000
0.050     8b       -43m
   2 Al   Al          0.12500    0.12500    0.12500    1.000
0.050    16c       .-3m
   3 O    O           0.36000    0.36000    0.36000    1.000
0.050    32e        .3m
==================================================================
```

Листинг

```
import numpy as np
import matplotlib.pyplot as plt
from collections import deque

plt.rcParams["figure.figsize"] = (3, 3)
%matplotlib notebook

asymmetric_unit_dict = {}
symmetry_operations_dict = {}
wyckoff_positions_dict = {}
with open('CUBIC.DAT') as f:
    while True:
        group_name = f.readline().strip()
        if group_name == 'HALT':
            break
        n = int(f.readline())
        asymmetric_unit = []
        for i in range(n):
            a, b, c, d = (float(t) for t in f.readline().split())
            asymmetric_unit.append((np.array([a, b, c]),d))
        back_symmetry_operations = []
        for i in range(n):
            syms, *tail = f.readline().split()
            dx, dy, dz, denom = (float(t) for t in tail)
            matrix = np.zeros((3, 3))
            for i in range(3):
                mult = 1 if syms[i].islower() else -1
                ptr = ord(syms[i].lower()) - ord('x')
                matrix[i][ptr] = mult
            dd = np.array([dx, dy, dz]) / denom
            back_symmetry_operations.append((matrix, dd))
        m = int(f.readline())
        symmetry_operations = []
        for i in range(m):
            syms, *tail = f.readline().split()
            dx, dy, dz, denom = (float(t) for t in tail)
            matrix = np.zeros((3, 3))
            for i in range(3):
                mult = 1 if syms[i].islower() else -1
                ptr = ord(syms[i].lower()) - ord('x')
                matrix[i][ptr] = mult
            dd = np.array([dx, dy, dz]) / denom
            symmetry_operations.append((matrix, dd))
        k = int(f.readline())
        wyckoff_positions = {}
        for i in range(k):
            name, mult, dims, *tail = f.readline().split()
            x, y, z, denom, *freedoms = (float(t) for t in tail)
            pos = np.array([x, y, z]) / denom
            wyckoff_positions[name] = (int(mult), pos, [np.array(freedoms[j:j+3]) / denom for j in
range(int(dims))])
        asymmetric_unit_dict[group_name] = asymmetric_unit
        symmetry_operations_dict[group_name] = symmetry_operations
        wyckoff_positions_dict[group_name] = wyckoff_positions

def build_atoms(group, base_positions, free_variables):
    wyckoff_positions = wyckoff_positions_dict[group]
    symmetry_operations = symmetry_operations_dict[group]
    all_atoms = []
    var_ptr = 0
    for (pos, name, radius) in base_positions:
        local_atoms = []
        actual_positions = wyckoff_positions[pos][1].copy()
        for (i, add) in enumerate(wyckoff_positions[pos][2]):
            actual_positions += free_variables[var_ptr * i] * add
        var_ptr += len(wyckoff_positions[pos][2])
        q = deque()
        q.append(actual_positions)
        local_atoms.append((actual_positions, name, radius))
        while q:
            cur_pos = q.popleft()
            for (matrix, add) in symmetry_operations:
                next_pos = np.remainder(np.dot(matrix, cur_pos) + add, 1)
                is_new = True
                for (other_pos, other_name, other_radius) in local_atoms:
                    distance = np.linalg.norm(np.remainder(next_pos - other_pos + 0.5, 1) - 0.5)
                    is_new = is_new and distance > 1e-8
                if is_new:
                    local_atoms.append((next_pos, name, radius))
                    q.append(next_pos)
        all_atoms += local_atoms
    return all_atoms

group_name = 'Fd3m'
base_positions = [
    ('b', 'Fe', 0.74),
    ('c', 'Al', 0.51),
    ('e', 'O', 1.32),
]
low_a = 3
high_a = 10
best_vars = []
penalties = []
var_steps = 1000

free_variables_count = 0
total_atoms_count = 0
min_len = 1000
for (pos, name, radius) in base_positions:
    free_variables_count += len(wyckoff_positions_dict[group_name][pos][2])
    for direction in wyckoff_positions_dict[group_name][pos][2]:
        min_len = min(min_len, np.linalg.norm(direction))
    total_atoms_count += wyckoff_positions_dict[group_name][pos][0]

for step in range(10):
    real_a = (low_a + high_a) * 0.5
    lowest = 1e10
    res = 0
    skip_to = 0
    penalties = []
    for ptr in range(var_steps + 1):
        if ptr < skip_to:
            continue
        all_atoms = build_atoms(group_name, base_positions, [ptr / var_steps])
        if len(all_atoms) != total_atoms_count:
            continue
        penalty = 0
        for i in range(len(all_atoms)):
            for j in range(i):
                pos_i = all_atoms[i][0]
                pos_j = all_atoms[j][0]
                radius_i = all_atoms[i][2]
                radius_j = all_atoms[j][2]
                distance = real_a * np.linalg.norm(np.remainder(pos_i - pos_j + 0.5, 1) - 0.5)
                penalty = max(penalty, radius_i + radius_j - distance)
        if penalty < lowest:
            lowest = penalty
            res = ptr
            if lowest == 0:
                break
        skip_to = ptr + penalty / real_a / min_len / 2 * var_steps
    print(step, real_a, lowest, res / var_steps)
    best_vars = [res / var_steps]
    if lowest == 0:
        high_a = real_a
    else:
        low_a = real_a

print(low_a, high_a)

all_atoms = build_atoms(group_name, base_positions, best_vars)
for atom in all_atoms:
    print(atom)
print(len(all_atoms))

xs = [pos[0] for (pos, name, radius) in all_atoms]
ys = [pos[1] for (pos, name, radius) in all_atoms]
zs = [pos[2] for (pos, name, radius) in all_atoms]
color = [(0 if name == 'Fe' else (1 if name == 'Al' else 2)) for (pos, name, radius) in all_atoms]
fig = plt.figure()
ax = fig.add_subplot(projection = '3d')
ax.scatter(xs, ys, zs, c = color)
ax.set_xlim(0, 1)
ax.set_ylim(0, 1)
ax.set_zlim(0, 1)
ax.set_xlabel('X Axis')
ax.set_ylabel('Y Axis')
ax.set_zlabel('Z Axis')
plt.show()
```