

Молекулярная динамика

Этап №3

Гафиров Абдималик НФИбд-01-18; Логинов Сергей НФИбд-01-18;
Мулихин Павел НФИбд-01-18; Наливайко Сергей НФИбд-01-18;
Смирнова Мария НФИбд-01-18; Сорокин Андрей НФИбд-03-18

Ход работы

Для примера возьмем атом меди со следующими начальными условиями

$$N = 100$$

$$a = 2.5974 * 10^{-10}$$

$$b = 1.1956 * 10^{-20}$$

$$m = 1.07 * 10^{-27}$$

Составим программный код, который будет моделировать поведение атомов меди.

Программный код

```
1 class Atom:
2     def __init__(self, data):
3         self.x0 = data[0]
4         self.y0 = data[1]
5         self.x1 = data[0]
6         self.y1 = data[1]
7         self.v1x = 0.0
8         self.v1y = 0.0
9         self.v0x = 0.0
10        self.v0y = 0.0
11        self.xt = data[0]
12        self.yt = data[1]
13        self.f = 0
14
```

Рис. 1: Класс атома

```
22 def generate_data(n, a):  
23     data = []  
24     d = a * 1.3919  
25     for i in range(n):  
26         ppp = i / 10.0  
27         nn = i % 10.0  
28         x = d * nn  
29         y = d * ppp  
30         data.append(Atom([x, y]))  
31     return data  
32
```

Рис. 2: Генерация массива атомов для элемента

```
N = 100
a = 2.5974 * pow(10, -10)
b = 1.1956 * pow(10, -20)
m = 1.07 * pow(10, -27)
dt = 5 * pow(10, -100)

atoms = generate_data(N, a)
```

Рис. 3: Задание начальных условий и генерация данных

```
66 for k in range(1, 20):
67     for i in range(N):
68         atoms[i].f = 0
69         for j in range(N):
70             if i != j:
71                 atoms[i].f += potential_ld(atoms[i], atoms[j], a, b)
72         if k == 1:
73             atoms[i].v1x = atoms[i].v0x + (atoms[i].f / m) * dt
74             atoms[i].v1y = atoms[i].v0y + (atoms[i].f / m) * dt
75             atoms[i].x1 = atoms[i].x0 + atoms[i].v1x * dt
76             atoms[i].y1 = atoms[i].y0 + atoms[i].v1y * dt
77             atoms[i].v0x = atoms[i].v1x
78             atoms[i].v0y = atoms[i].v1y
79         else:
80             atoms[i].xt = 2 * atoms[i].x1 - atoms[i].x0 + atoms[i].f / m * pow(atoms[i].x1, 2)
81             atoms[i].yt = 2 * atoms[i].y1 - atoms[i].y0 + atoms[i].f / m * pow(atoms[i].y1, 2)
82
83             atoms[i].x0 = atoms[i].x1
84             atoms[i].x1 = atoms[i].xt
85             atoms[i].y0 = atoms[i].y1
86             atoms[i].y1 = atoms[i].yt
87
88 save_plot(atoms, k)
```

Рис. 4: Цикл решения


```
9 def save_plot(a, count):
10     f, ax = plt.subplots(1)
11     for i in range(len(a)):
12         ax.plot(a[i].x0, a[i].y0, '.', color='green')
13     plt.savefig('plots/img{}.png'.format(count))
```

Рис. 5: Сохранение результатов

Результат

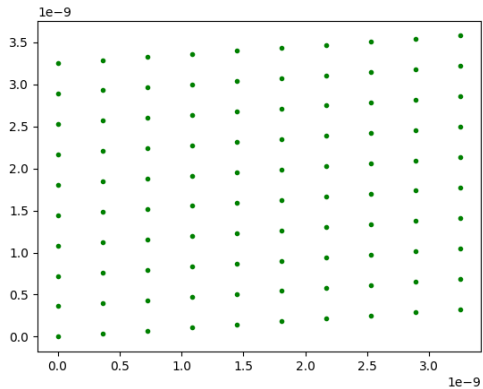


Рис. 6: Состояние 1

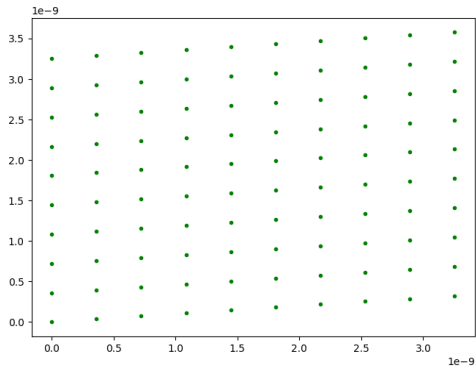


Рис. 7: Состояние 2

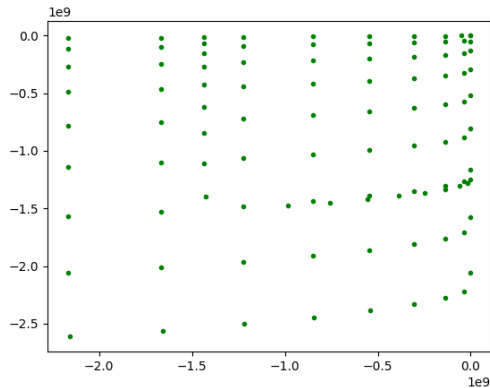


Рис. 8: Состояние 3

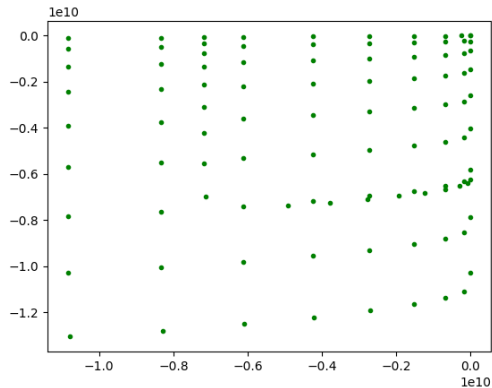


Рис. 9: Состояние 4

Вывод

В ходе третьего этапа проекта мы смоделировали процесс двумерной молекулярной динамики