

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

Общие сведения

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

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
```

Рис. 1: Класс атома

Программный код

```
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
32     return data
```

Рис. 2: Генерация массива атомов для элемента

Программный код

```
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

Программный код

```
46     for k in range(i, 20):
47         for i in range(N):
48             atoms[i].r = 0
49             for j in range(N):
50                 if i != j:
51                     atoms[i].r += potential_ld(atoms[i], atoms[j], a, b)
52             if k == 1:
53                 atoms[i].vix = atoms[i].v0x + (atoms[i].r / n) * dt
54                 atoms[i].viy = atoms[i].v0y + (atoms[i].r / n) * dt
55                 atoms[i].x1 = atoms[i].x0 + atoms[i].vix * dt
56                 atoms[i].y1 = atoms[i].y0 + atoms[i].viy * dt
57                 atoms[i].v0x = atoms[i].vix
58                 atoms[i].v0y = atoms[i].viy
59             else:
60                 atoms[i].xt = 2 * atoms[i].x1 - atoms[i].x0 + atoms[i].r / n * pow(atoms[i].x1, 2)
61                 atoms[i].yt = 2 * atoms[i].y1 - atoms[i].y0 + atoms[i].r / n * pow(atoms[i].y1, 2)
62
63                 atoms[i].x0 = atoms[i].x1
64                 atoms[i].x1 = atoms[i].xt
65                 atoms[i].y0 = atoms[i].y1
66                 atoms[i].y1 = atoms[i].yt
67             save_plot(atoms, k)
```

Рис. 4: Цикл решения

Программный код

```
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

Результат

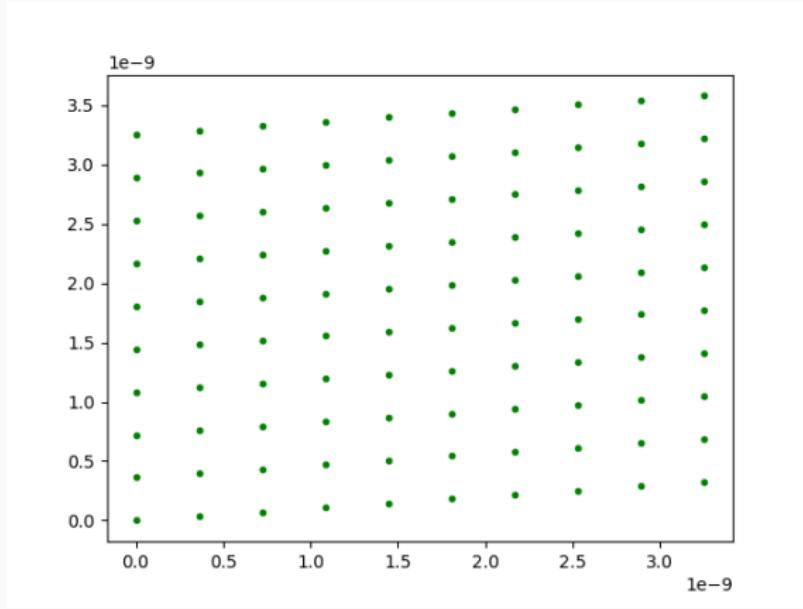


Рис. 6: Состояние 1

Результат

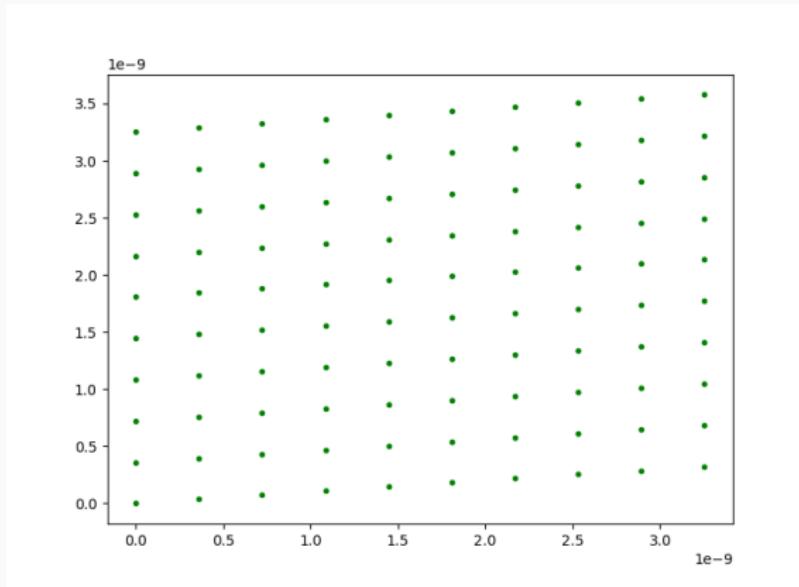


Рис. 7: Состояние 2

Результат

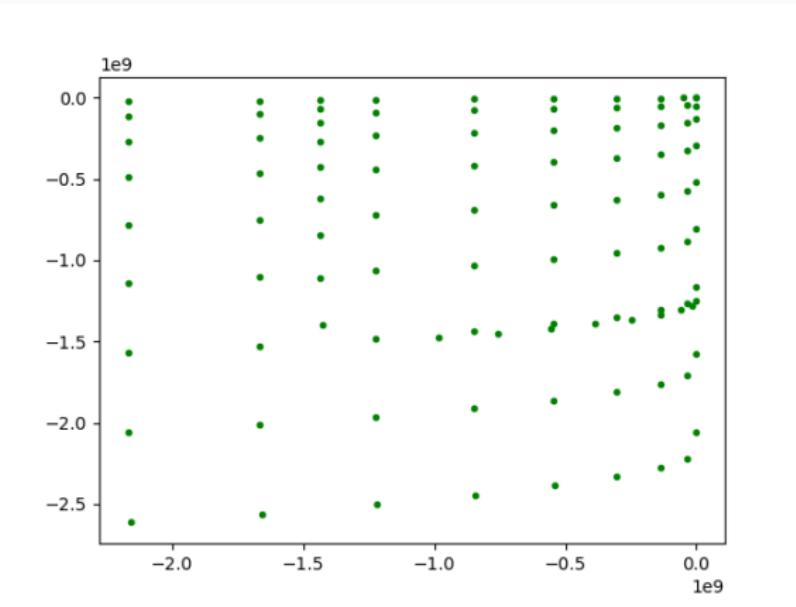


Рис. 8: Состояние 3

Результат

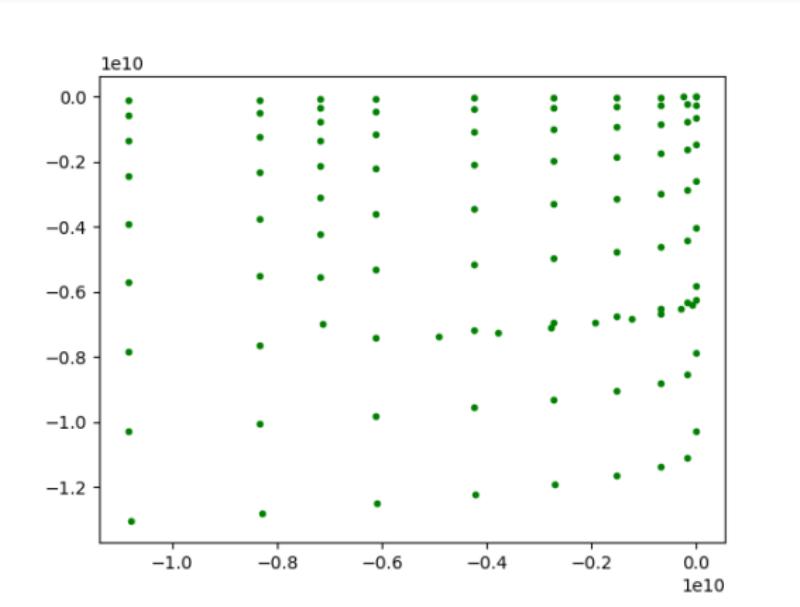


Рис. 9: Состояние 4

Вывод

Вывод

В ходе третьего этапа проекта мы смоделировали процесс двумерной молекулярной динамики