Проект по курсу “Численные методы в физике космической плазмы” на тему

"Моделирование движения заряженных частиц в электромагнитном поле. Сравнение прямых и параллельных вычислений".

Московский Физико-Технический Институт
Физтех-школа физики и исследований им. Ландау
Студент 782 группы: Сатыбалдиев М.Н.
Преподаватель: Попов В.Ю.
Декабрь 2020

Введение

В данной работе рассмотрено моделирование движения заряженных частиц в однородном электромагнитном поле. Также были сравнены времена линейных, многопроцессорных и многопоточных вычислений на языке программирования Python.

Динамика частицы

Уравнение движения частицы массы m заряда q в электромагнитном поле $\vec{E}, \vec{H}$ выглядит следующим образом:

$$\begin{cases} \dfrac{d\vec{V}}{dt} = \dfrac{q}{m}(\vec{E} + \vec{V} \times \vec{B}), \\ \vec{V} = \dfrac{d\vec{R}}{dt}, \end{cases} \tag{1}$$

где $\vec{R}$ - радиус-вектор частицы, $\vec{V}$ - ее скорость.

Будем рассматривать движение в однородном поле с $\vec{E} = \begin{pmatrix} E \\ 0 \\ 0 \end{pmatrix}$ и $\vec{H} = \begin{pmatrix} H \\ 0 \\ 0 \end{pmatrix}$.
В таком поле будет наблюдаться ускорение частицы вдоль оси x и вращение в плоскости yz с частотой $\Omega_c = \dfrac{qH}{m}$.

Разностная схема

Для решения задачи была использована схема с перешагиванием(*Leap-Frog Method*):

$$\begin{cases} \vec{R}_n = \vec{R}_{n-1} + \vec{V}_{n-\frac{1}{2}}\Delta t \\ \vec{V}_{n+\frac{1}{2}} = \vec{V}_{n-\frac{1}{2}} + \dfrac{q}{m}(\vec{E}_{n-\frac{1}{2}} + \vec{V}_{n-\frac{1}{2}} \times \vec{H}_{n-\frac{1}{2}})\Delta t \end{cases} \tag{2}$$

В упрощенном виде[2] второе уравнение принимает вид:

$$\vec{V}_{n+\frac{1}{2}} = \vec{V}_{n-\frac{1}{2}}(1 - \frac{1}{2}\Omega_c^2\Delta t^2) + \frac{q\Delta t}{m}(\vec{E} + \vec{V}_{n-\frac{1}{2}} \times \vec{H}) + \frac{q^2\Delta t^2}{2m^2}(\vec{E} \times \vec{H}) + \frac{q^2\Delta t^2}{2m^2}(\vec{V}_{n-\frac{1}{2}\cdot H})\vec{H} \tag{3}$$

Данная схема является обратимой по времени и обладает точностью порядка $\mathcal{O}(\Delta t^3)$.

Проверка обратимости

Наглядным методом проверки правильности работы метода является обращение времени. Пусть для некоторой частицы было сделано N шагов вычислений. Тогда новая частица с начальной координатой, равной конечной координате первой, и противоположно направленной скоростью через N шагов должна оказаться в начальной точке первой частицы.

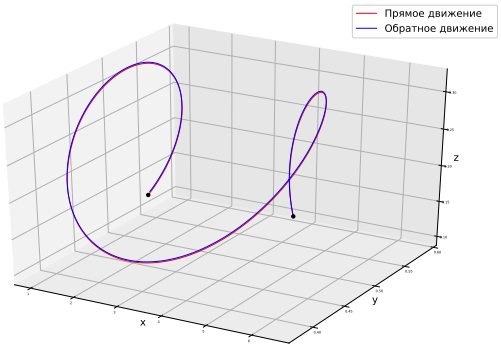

Рис. 1: Красная линия - прямое во времени движение частицы, синее - обратное.

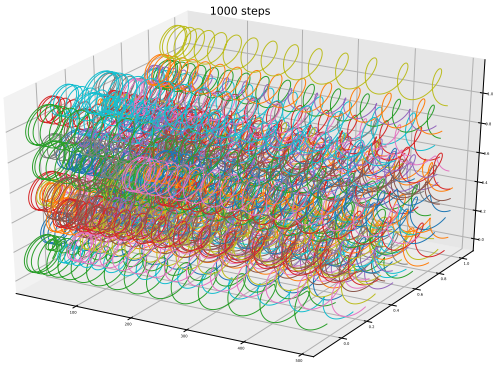

Рис. 2: Траектории 100 частиц.

Параллельные вычисления

В рассматриваемой модели не учитывались взаимные взаимодейсвия частиц, поэтому данная задача позволяет проводить параллельные вычисления, причем асинхронно.

Для распараллеливания использовалась стандартная библиотека Python *concurrent.futures*. В частности использовались *ProcessPoolExecutor* и *ThreadPoolExecutor* ([3]).

ProcessPoolExecutor используется для мультипроцессорных вычислений. То есть выполнение программы происходит на нескольких ядрах процессора параллельно. *ThreadPoolExecutor* разделяет задачу в одном процессе на несколько потоков, которые выполняются параллельно.

Ожидается, что время работы программы значительно уменьшится при мультипроцессорных вычислениях траекторий множества частиц.

Ниже представлены графики со сравнением времени выполнения от числа частиц и шагов (вычисления производились на комьютере с 8 Гб ОЗУ и процессором Intel 2.4 GHz Core i5), таблицы со значениями см. в приложении.

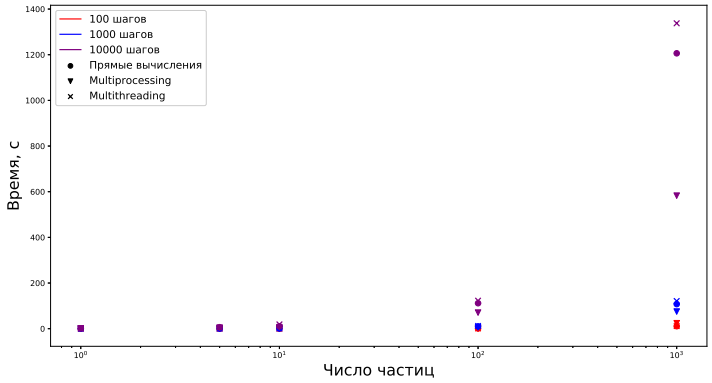

Рис. 3: Время выполнения в зависимости от числа частиц и шагов.

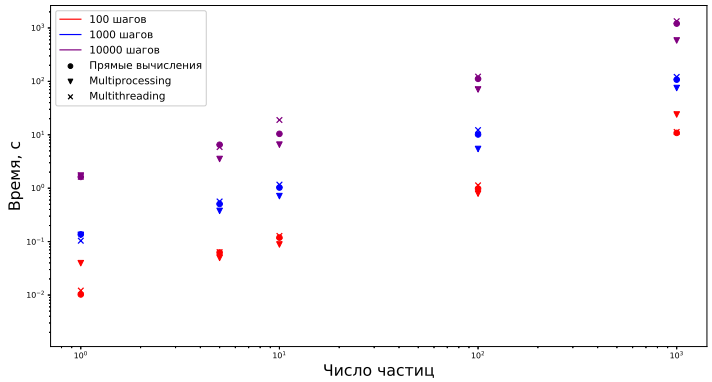

Рис. 4: В логарифмическом масштабе.

Видно, что при большом числе частиц и шагов multiprocessing обладает большим преимуществом. Multithreading же наоборот работает медленно. Это связано с особенностями языка программирования Python (язык интерпретируемый, а не компилируемый), а именно с GIL (Global interpreter lock).

Итоги

В этой работе была реализована Leap-Frog схема решения уравнений движения заряженной частицы в поле, была добавлена параллельность вычислений в нескольких процессах и потоках.

Также было показано преимущество параллельных расчетов в данной задаче. Программа позволяет проводить расчеты не только в однородных полях, но и в неоднородных по пространству и времени.

Приложение

Таблица 1: 100 шагов

Число частиц	Прямой метод	Multiprocessing	Multithreading
1	0.010246038436889648	0.039484262466430664	0.012104988098144531
5	0.06265115737915039	0.04951333999633789	0.06361007690429688
10	0.11876821517944336	0.08832788467407227	0.12803888320922852
100	0.9815783500671387	0.7908670902252197	1.1305022239685059
1000	10.849240064620972	23.966799020767212	11.323905229568481

Таблица 2: 1000 шагов

Число частиц	Прямой метод	Multiprocessing	Multithreading
1	0.13631820678710938	0.13383698463439941	0.10383796691894531
5	0.5069730281829834	0.3759758472442627	0.5637118816375732
10	1.028635025024414	0.7111492156982422	1.1619477272033691
100	10.129580020904541	5.422935962677002	12.259056806564331
1000	107.75596189498901	75.05907106399536	121.37761974334717

Таблица 3: 10000 шагов

Число частиц	Прямой метод	Multiprocessing	Multithreading
1	1.6266069412231445	1.7235348224639893	1.6336078643798828
5	6.512508153915405	3.5230300426483154	5.875111818313599
10	10.42299222946167	6.53804087638855	18.918434858322144
100	111.57286214828491	70.47390985488892	122.56676697731018
1000	1206.1796510219574	582.7200920581818	1337.8531761169434

Ссылки:
Код программы: https://github.com/maksatsat/particles-in-field
Обратимость по времени https://youtu.be/NS46SsDaTAk
Траектории 100 частиц https://youtu.be/z-mA1n3T_Z4
Движение 1000 частиц https://youtu.be/hF5w0CX8xbc

Литература

[1] S. Fatemi *Computing, Visualising and Analyzing Ion Trajectories* (2009)

[2] Ledvina et. al. *Modeling and Simulating Flowing Plasmas and Related Phenomena* (2008)

[3] https://docs.python.org/3/library/concurrent.futures.html