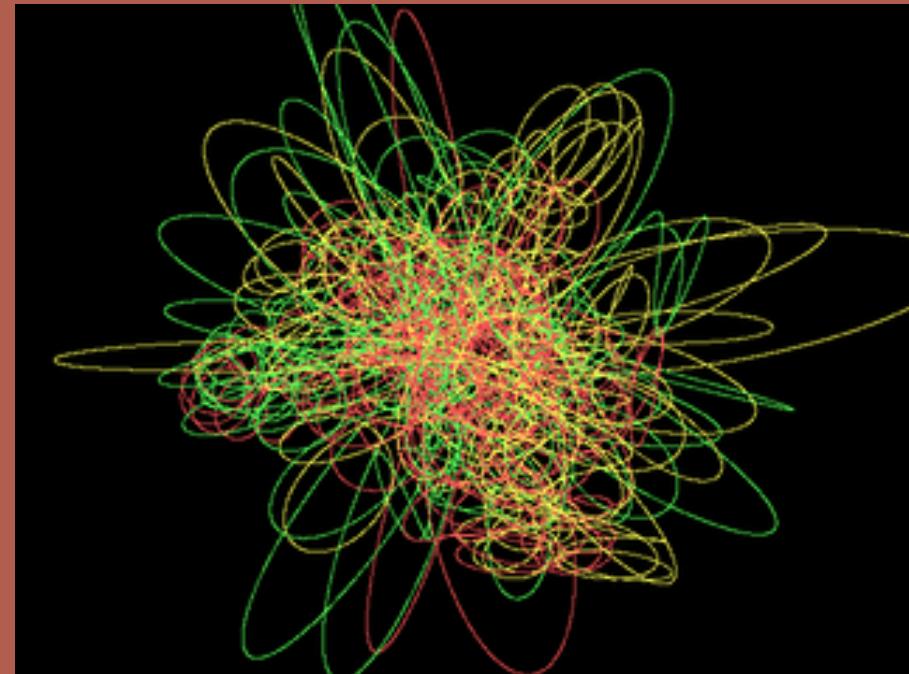


**Улучшение точности  
численного  
интегрирования  
гравитационной системы  
 $N$  тел с помощью  
сдвоенных чисел двойной  
точности**

Субботин Максим 9382

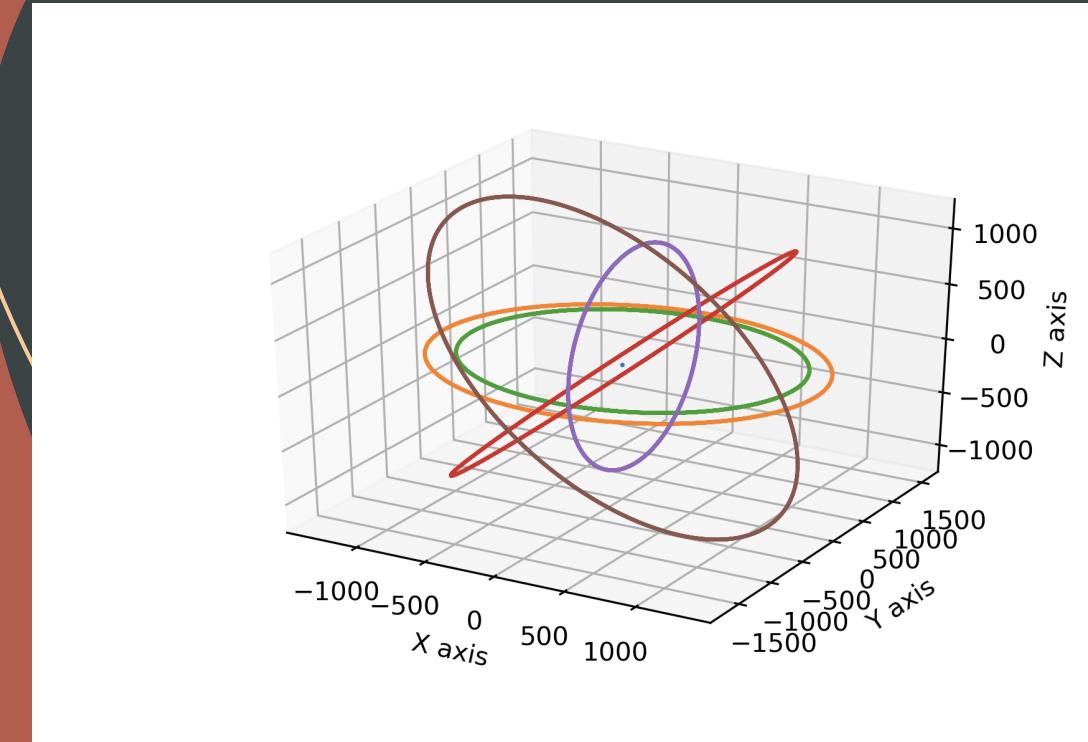
Кодуков Александр 9382



## Задача N тел

Описать эволюцию системы из N тел  
(материальных точек) в  
гравитационном поле.

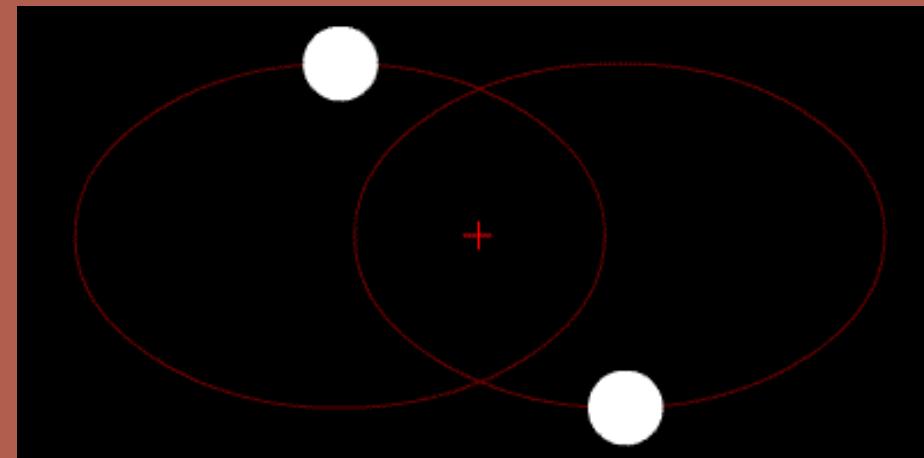
Тело задается массой, положением и  
скоростью.



# Задача N тел

- 1 тело: описывается первым законом Ньютона
- 2 тела: существует общее решение для нахождения орбиты
- 3+ тел: невозможно выразить через уравнения от координат и скоростей в общем случае.

Задача может быть решена только численными методами



# Физика явлений

- Сила гравитационного взаимодействия, действующая на тело  $n$ :

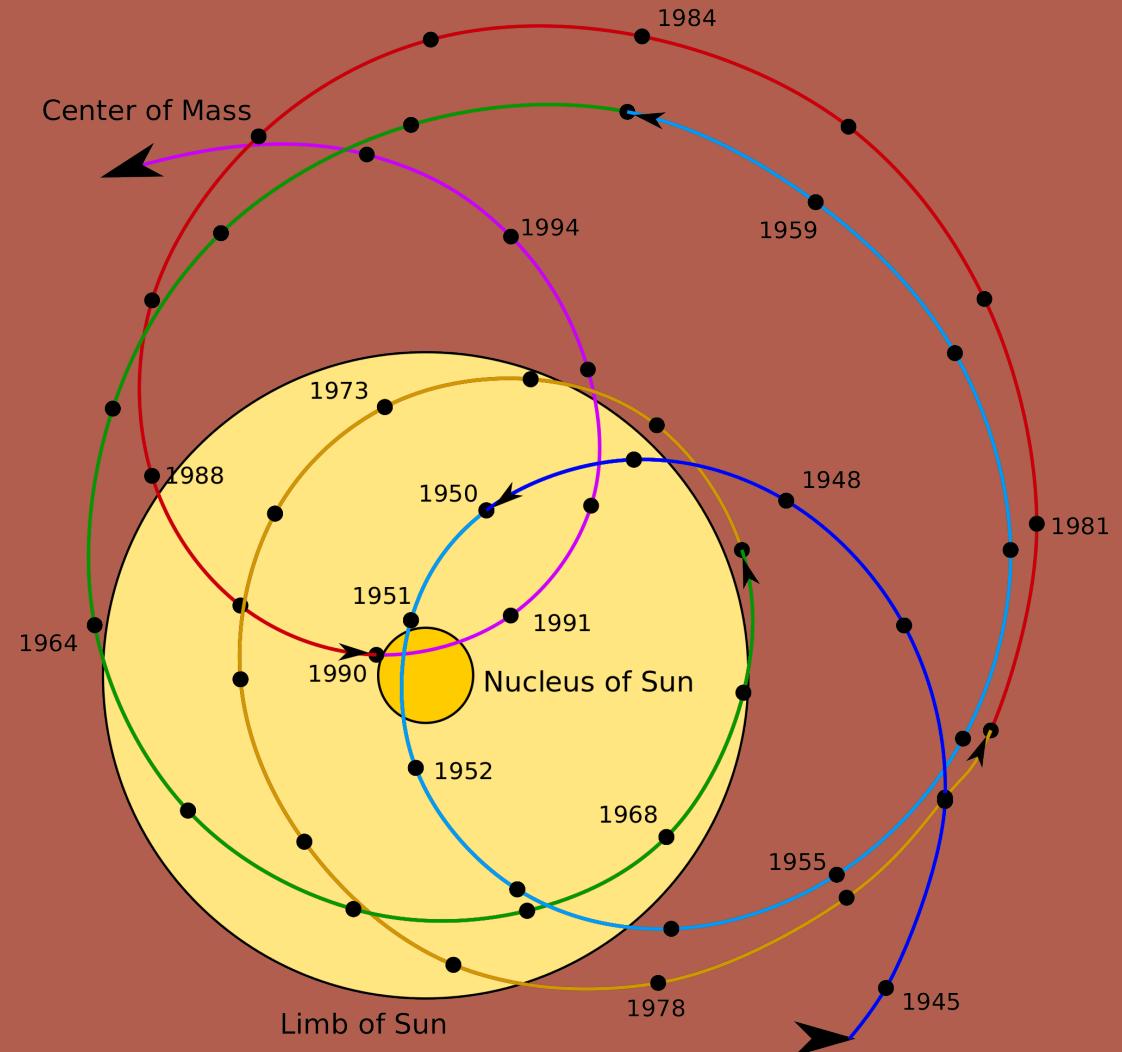
$$\vec{F}_n = -G \sum_{k \neq n} m_n m_k \frac{\vec{r}_n - \vec{r}_k}{|\vec{r}_n - \vec{r}_k|^3}$$

- Ускорение тела  $n$ :

$$\vec{a}_n = \vec{F}_n / m_n = -G \sum_{k \neq n} m_k \frac{\vec{r}_n - \vec{r}_k}{|\vec{r}_n - \vec{r}_k|^3}$$

# Инварианты

- Полная энергия системы
- Момент импульса системы
- Положение барицентра системы



# Задача Коши

Начальные данные:

$$\begin{cases} \frac{dy}{dx} = f(x, y) \\ y(x_0) = y_0 \end{cases}$$

Необходимо найти:

$$y(t) - ?$$

# Переход к численным методам

Переход к задаче первого порядка

$$\frac{\partial^2 \vec{r}_n}{\partial t^2} = a_n = -G \sum_{k \neq n} m_k \frac{\vec{r}_n - \vec{r}_k}{|\vec{r}_n - \vec{r}_k|^3}$$



$$\begin{aligned}\frac{\partial \vec{v}_n}{\partial t} &= f_n \\ \frac{\partial \vec{r}_n}{\partial t} &= \vec{v}_n\end{aligned}$$

# Семейство методов Рунге-Кутты

$$y_{n+1} = y_n + h \sum_{i=1}^s b_i k_i$$

$$k_1 = f(t_n, y_n),$$

$$k_2 = f(t_n + c_2 h, y_n + h(a_{21} k_1)),$$

$$k_3 = f(t_n + c_3 h, y_n + h(a_{31} k_1 + a_{32} k_2)),$$

⋮

$$k_i = f \left( t_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} k_j \right).$$

Таблица Бутчера:

$c_1$	$a_{11}$	$a_{12}$	⋮	$a_{1s}$
$c_2$	$a_{21}$	$a_{22}$	⋮	$a_{2s}$
⋮	⋮	⋮	⋮	⋮
$c_s$	$a_{s1}$	$a_{s2}$	⋮	$a_{ss}$
	$b_1$	$b_2$	⋮	$b_s$

# Метод Эйлера

$$y_{n+1} = y_n + h f(t_n, y_n).$$

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

Равноускоренное движение:

$$\vec{v}(t) = \vec{v}_0 + \vec{a}t$$

# Метод Рунге-Кутты 4 порядка

$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4),$$

$$t_{n+1} = t_n + h$$

$$k_1 = f(t_n, y_n),$$

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right),$$

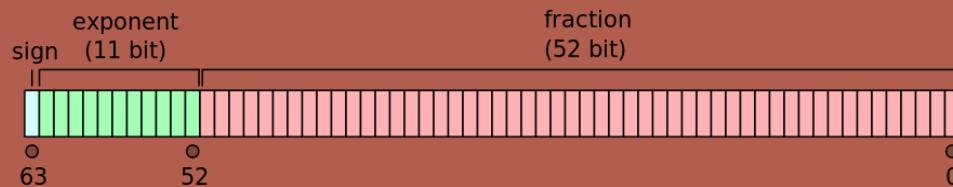
$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right),$$

$$k_4 = f(t_n + h, y_n + hk_3).$$

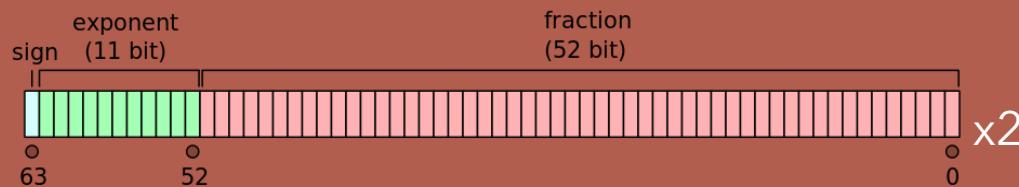
0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

# Используемые типы данных

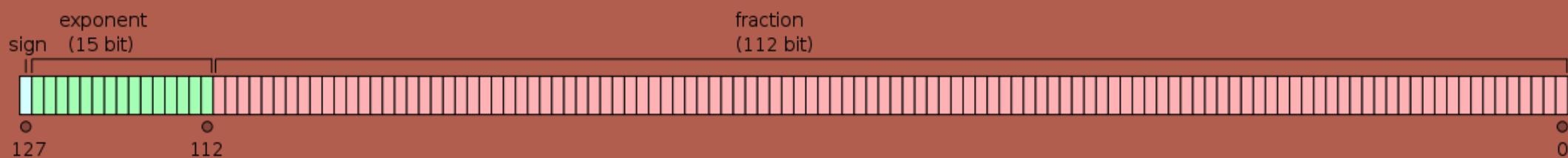
Double - число двойной точности



Double-double - сдвоенное число двойной точности



Long double - число четверной точности



# Арифметика double-double

Число в double-double:

$$x = x_l + x_h \text{ , где } |x_l| \leq \frac{1}{2} \text{ulp}(x_h)$$

Алгоритм суммирования с учетом ошибки:

---

**Algorithm 4.3** The Fast2Sum algorithm [108].

---

$s \leftarrow \text{RN}(a + b)$   
 $z \leftarrow \text{RN}(s - a)$   
 $t \leftarrow \text{RN}(b - z)$

---

Сложение чисел в double-double:

---

**Algorithm 14.1** Dekker's algorithm for adding two double-word numbers  $(x_h, x_\ell)$  and  $(y_h, y_\ell)$  [108]. We assume radix 2.

---

```
if  $|x_h| \geq |y_h|$  then
     $(r_h, r_\ell) \leftarrow \text{Fast2Sum}(x_h, y_h)$ 
     $s \leftarrow \text{RN}(\text{RN}(r_\ell + y_\ell) + x_\ell)$ 
else
     $(r_h, r_\ell) \leftarrow \text{Fast2Sum}(y_h, x_h)$ 
     $s \leftarrow \text{RN}(\text{RN}(r_\ell + x_\ell) + y_\ell)$ 
end if
 $(t_h, t_\ell) \leftarrow \text{Fast2Sum}(r_h, s)$ 
return  $(t_h, t_\ell)$ 
```

---

# Fused Multiply-Add/Subtract

$$\circ(ab + c)$$

Инструкция процессора, позволяющая посчитать  $a * b \pm c$ , с округлением только финального результата.

Пример использования:

---

**Algorithm 4.7** Dekker product.

**Require:**

$s = \lceil p/2 \rceil$

```
( $x_h, x_\ell$ )  $\leftarrow$  Split( $x, s$ )
( $y_h, y_\ell$ )  $\leftarrow$  Split( $y, s$ )
 $r_1 \leftarrow$  RN( $x \cdot y$ )
 $t_1 \leftarrow$  RN( $-r_1 + \text{RN}(x_h \cdot y_h)$ )
 $t_2 \leftarrow$  RN( $t_1 + \text{RN}(x_h \cdot y_\ell)$ )
 $t_3 \leftarrow$  RN( $t_2 + \text{RN}(x_\ell \cdot y_h)$ )
 $r_2 \leftarrow$  RN( $t_3 + \text{RN}(x_\ell \cdot y_\ell)$ )
```

---



---

**Algorithm 5.1** 2MultFMA( $x_1, x_2$ )

```
 $r_1 \leftarrow$  RN( $x_1 \cdot x_2$ )
 $r_2 \leftarrow$  RN( $x_1 \cdot x_2 - r_1$ )
```

---

# Алгоритмы суммирования

Алгоритм Кэхена:

```
function KahanSum(input)
    var sum = 0.0
    var c = 0.0

    for i = 1 to input.length do
        var y = input[i] - c
        var t = sum + y
        c = (t - sum) - y
        sum = t

    next i

return sum
```

Алгоритм Неймаера:

```
function NeumaierSum(input)
    var sum = 0.0
    var c = 0.0

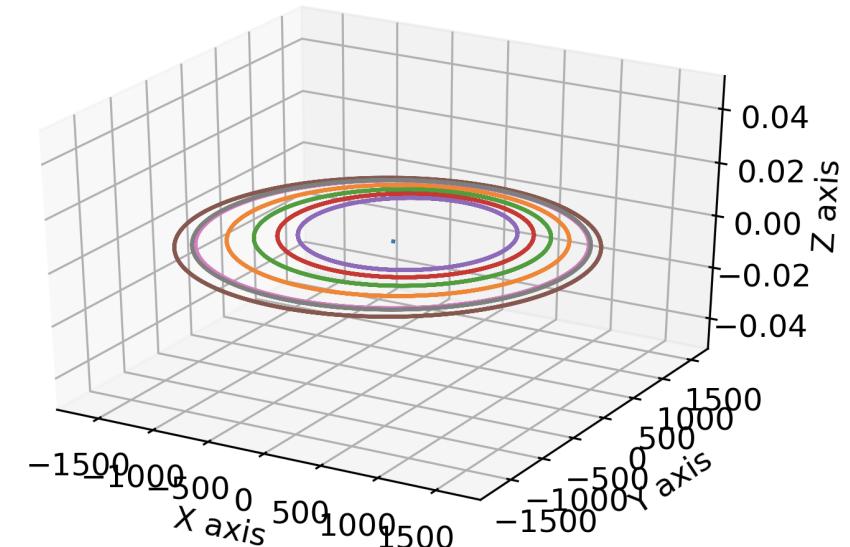
    for i = 1 to input.length do
        var t = sum + input[i]
        if |sum| >= |input[i]| then
            c += (sum - t) + input[i]
        else
            c += (input[i] - t) + sum
        endif
        sum = t

    next i

return sum + c
```

# Сравнение численного интегрирования на различных типах данных

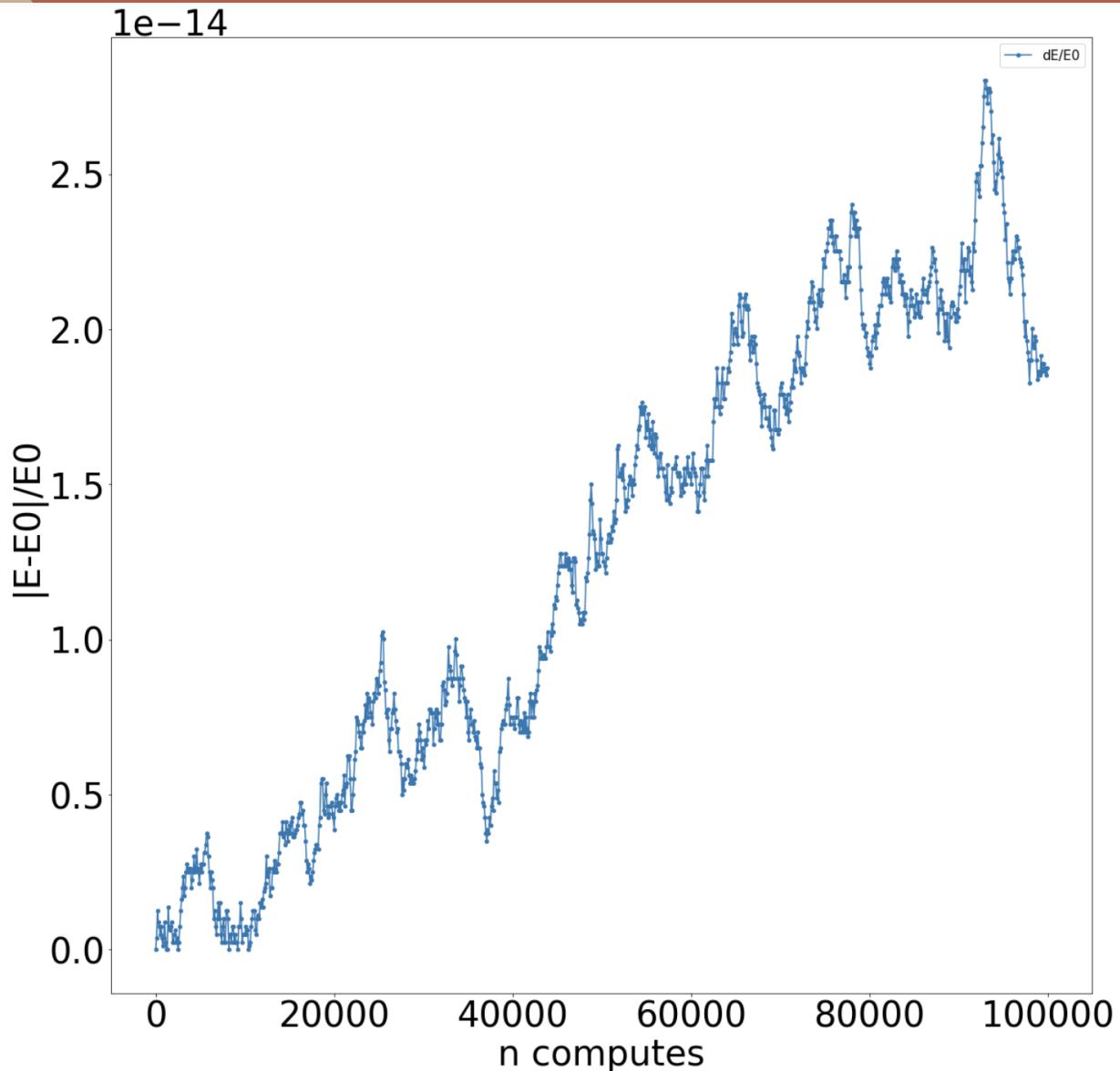
- Сравнение происходит на системе из 8 тел (планета и 7 спутников)
- $10^5$  итераций метода интегрирования
- С шагом 0.1
- Записываются каждые 100 значений



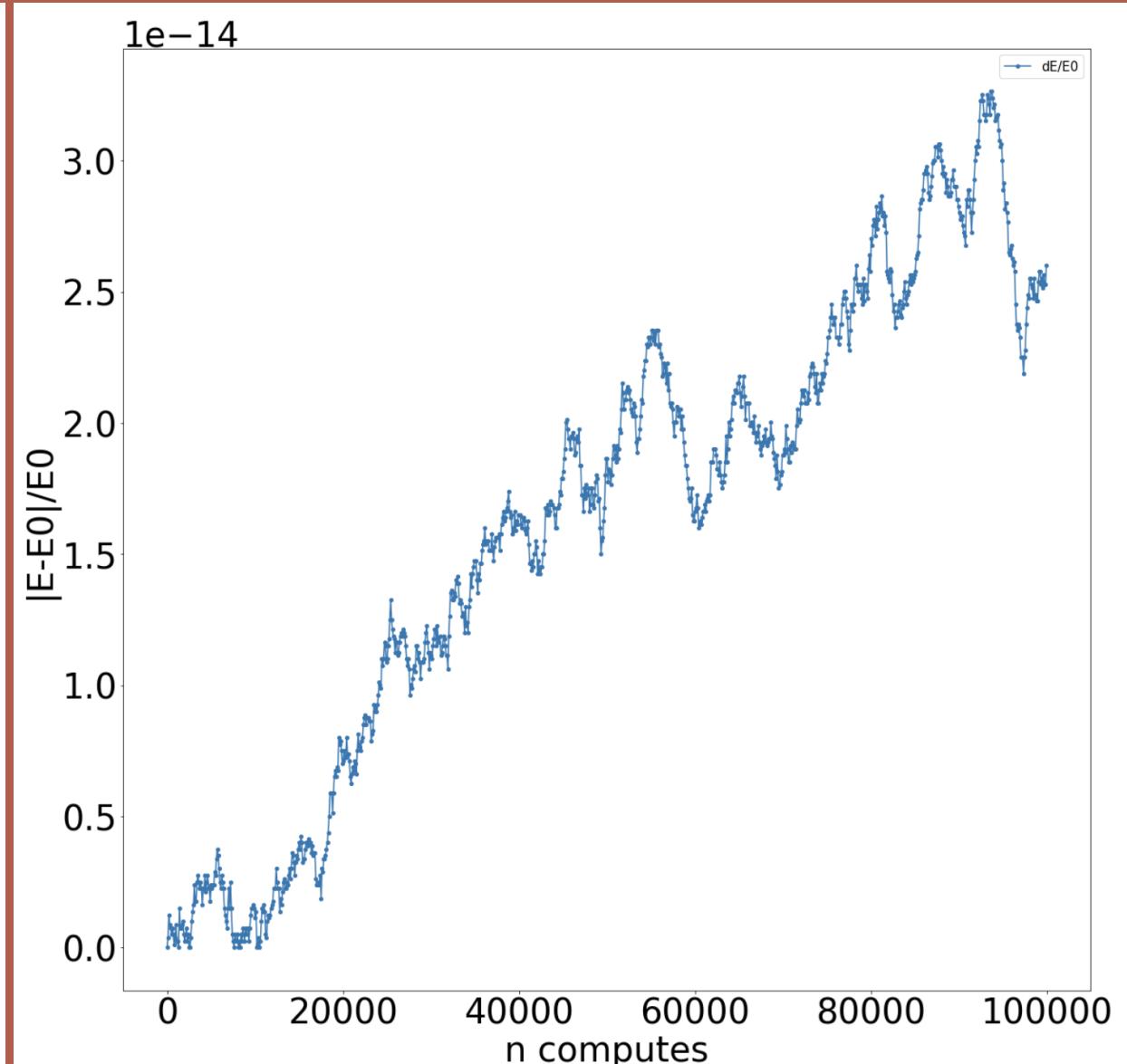
double(64 bits)

Энергия

RK4



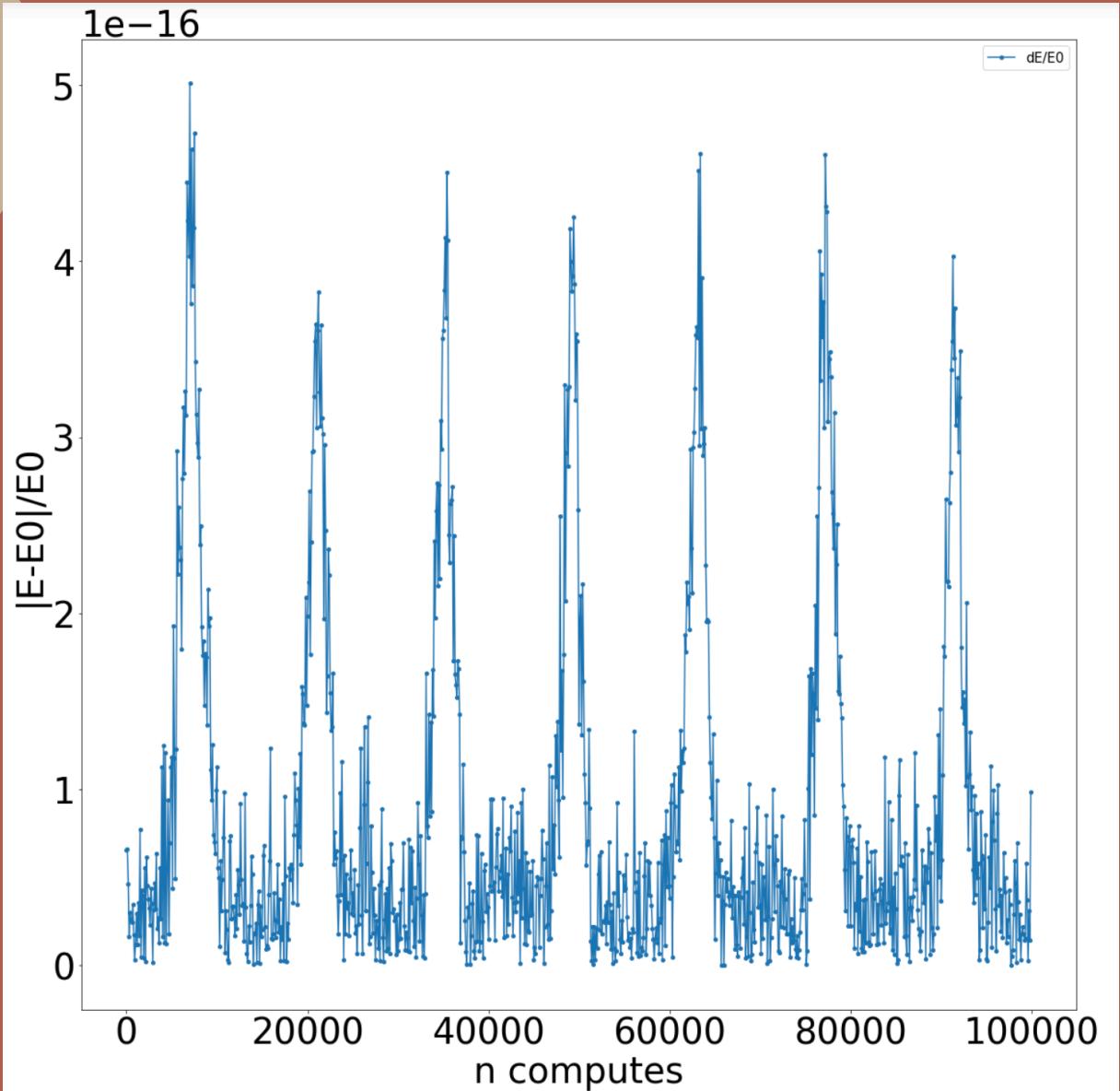
dopri8



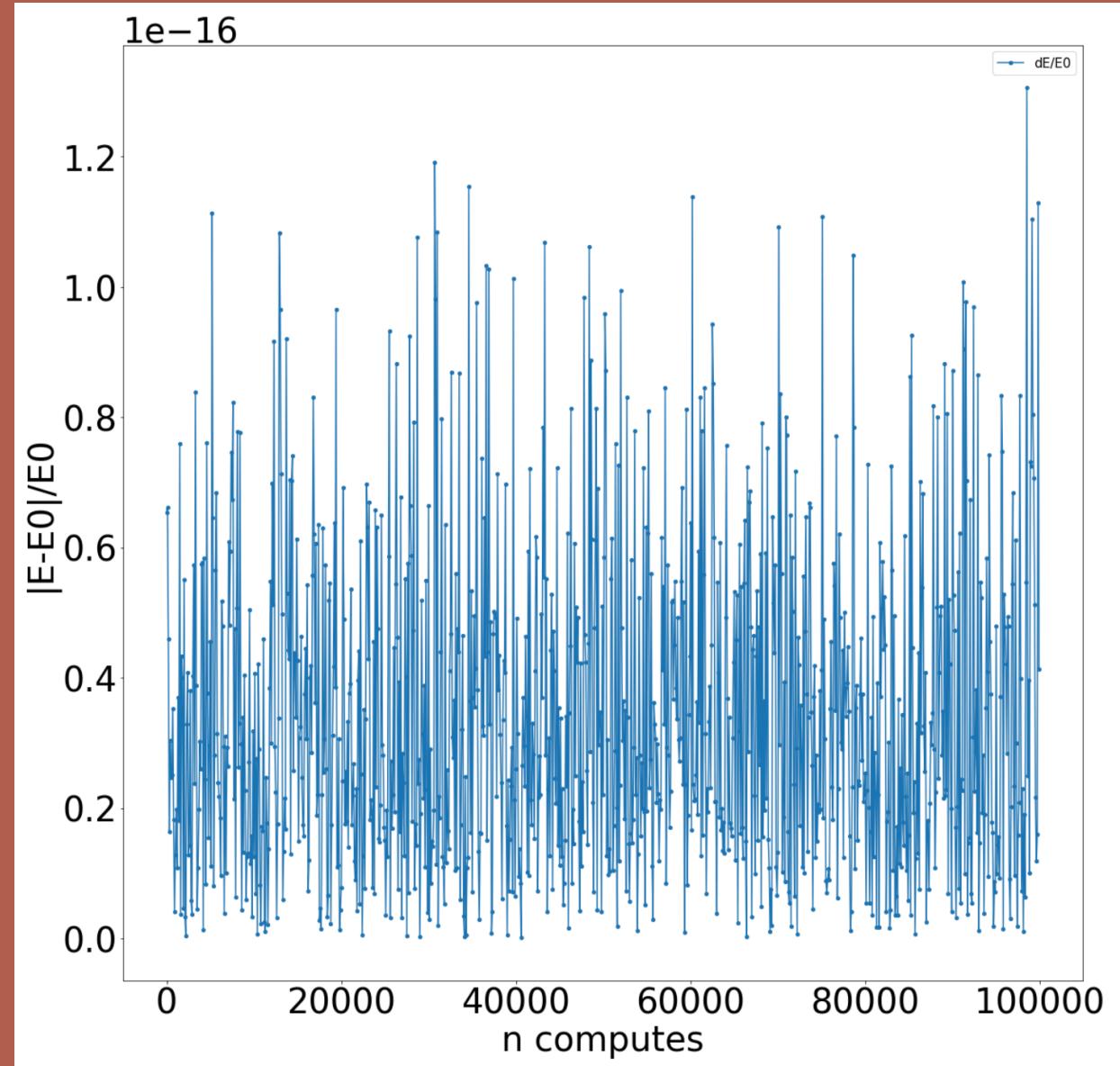
Long double(80 bits)

Энергия

RK4



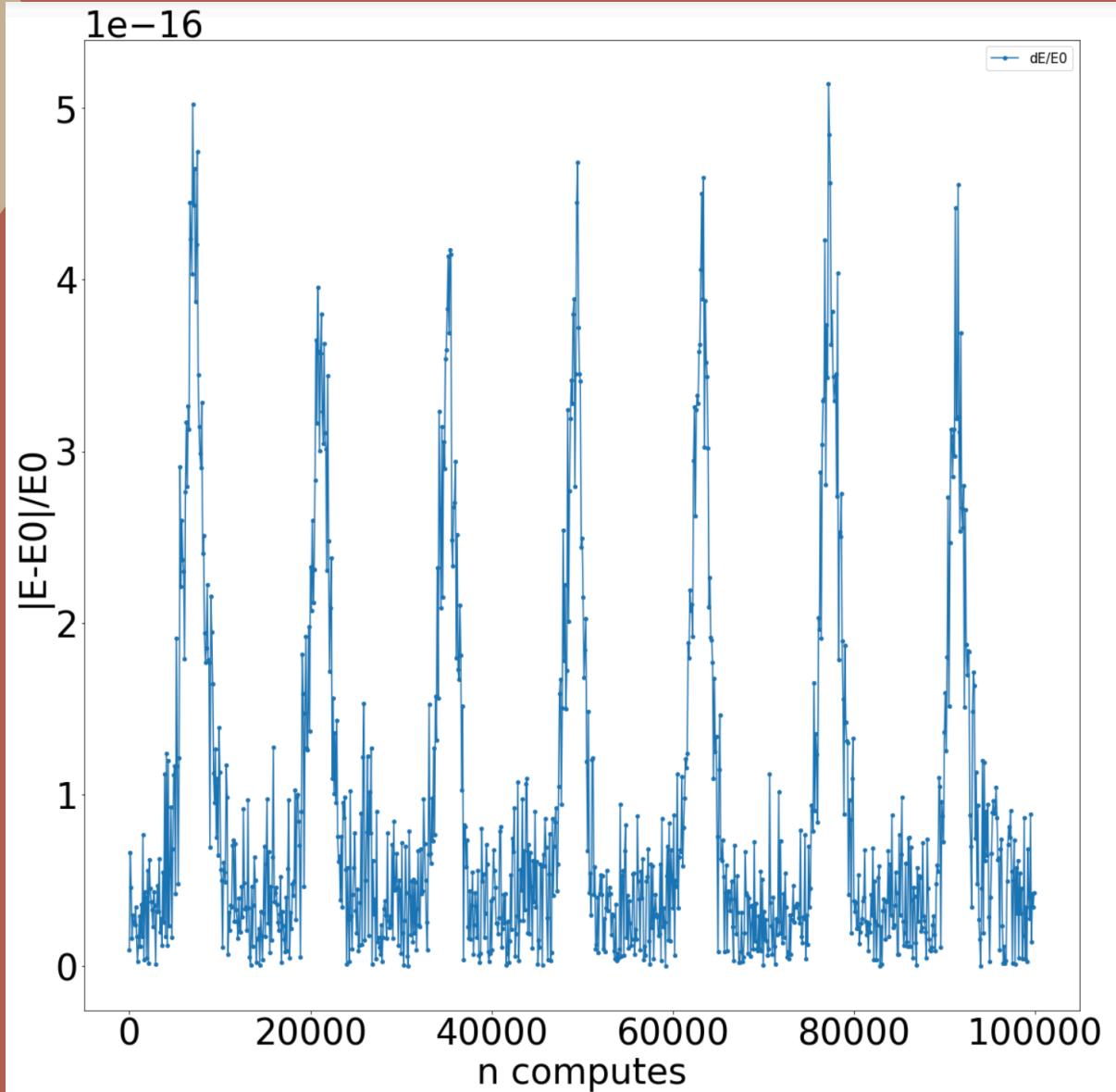
dopri8



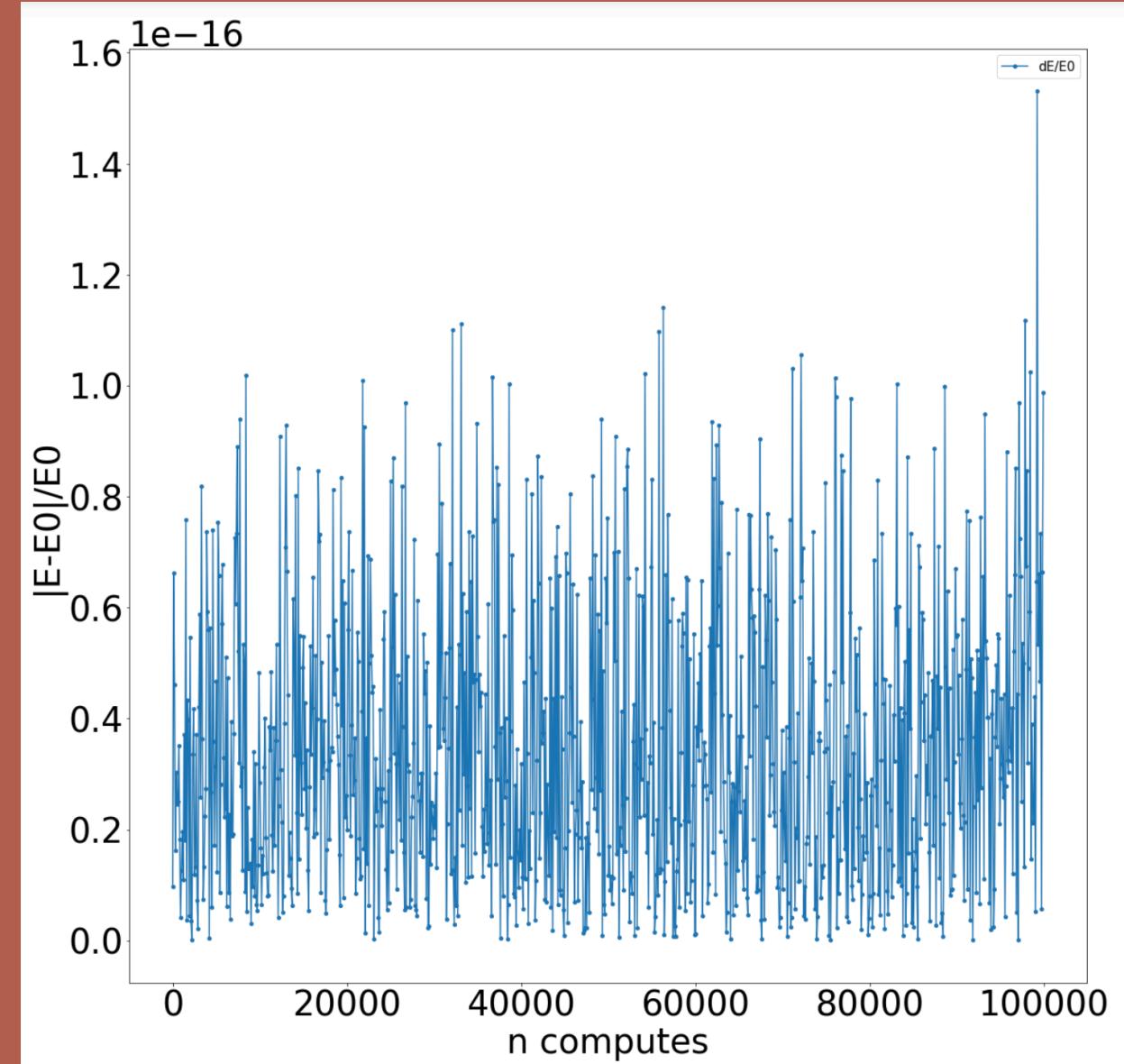
Long double(128 bits)

Энергия

RK4



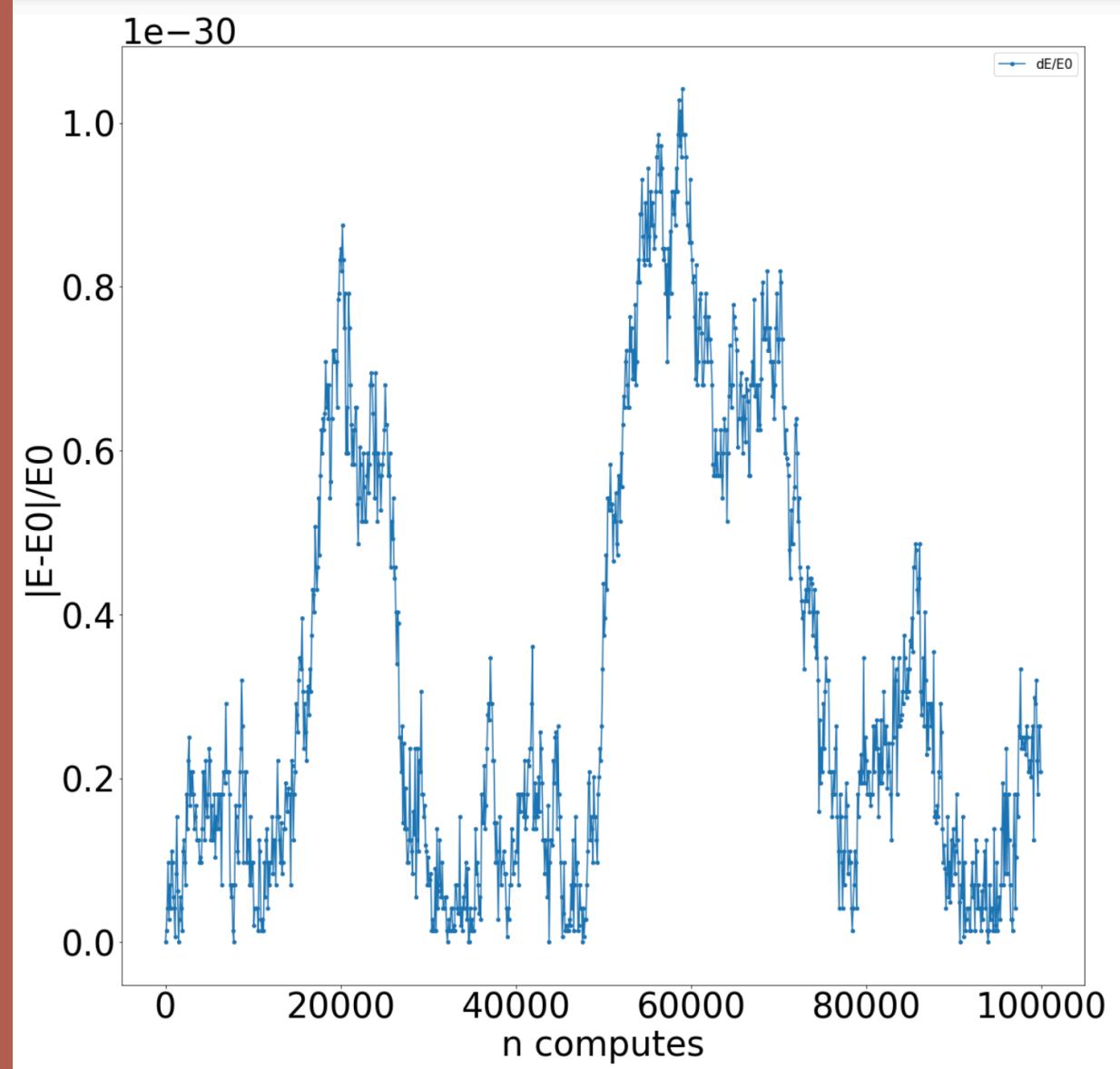
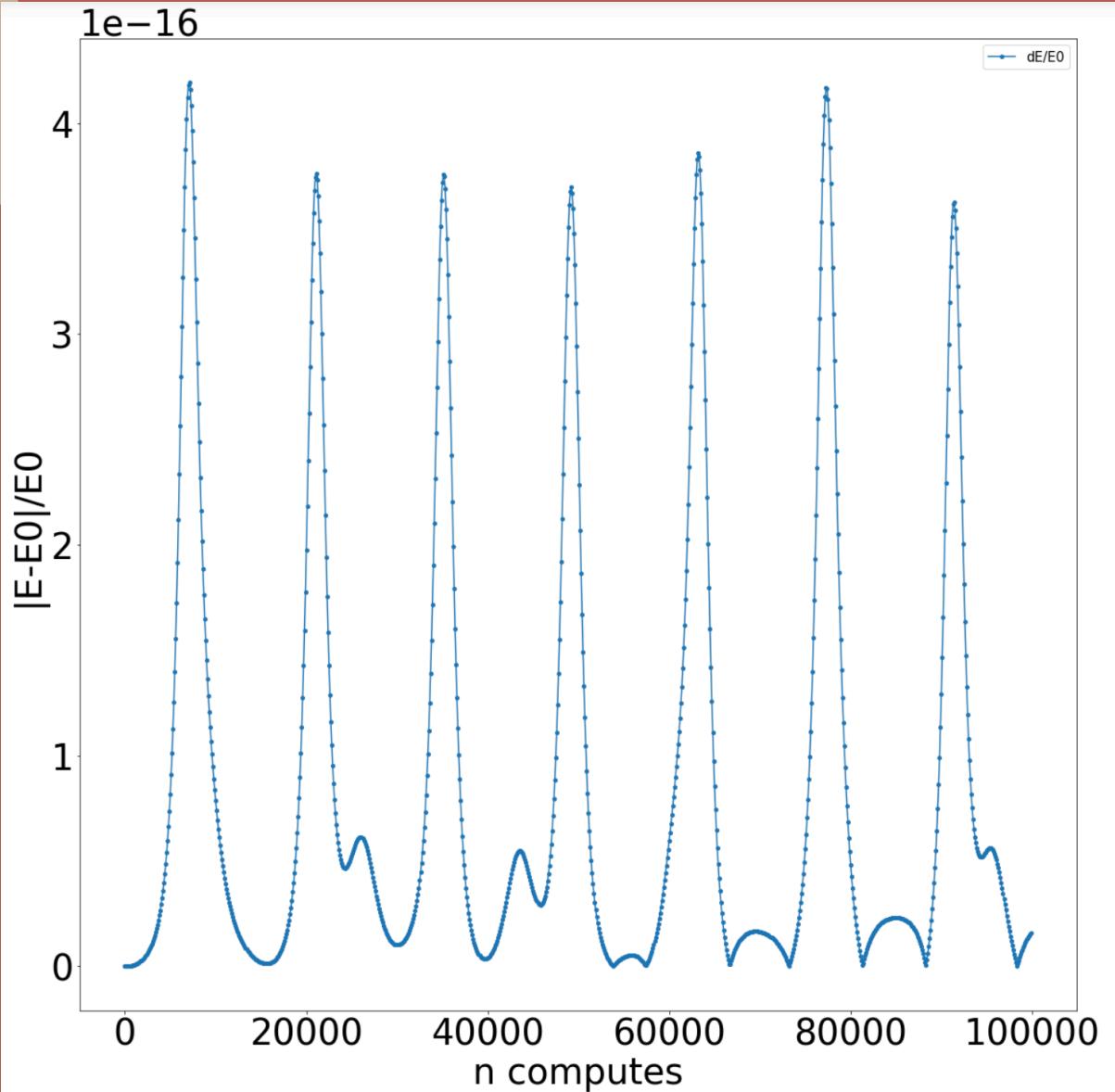
dopri8



Double-double  
Энергия

RK4

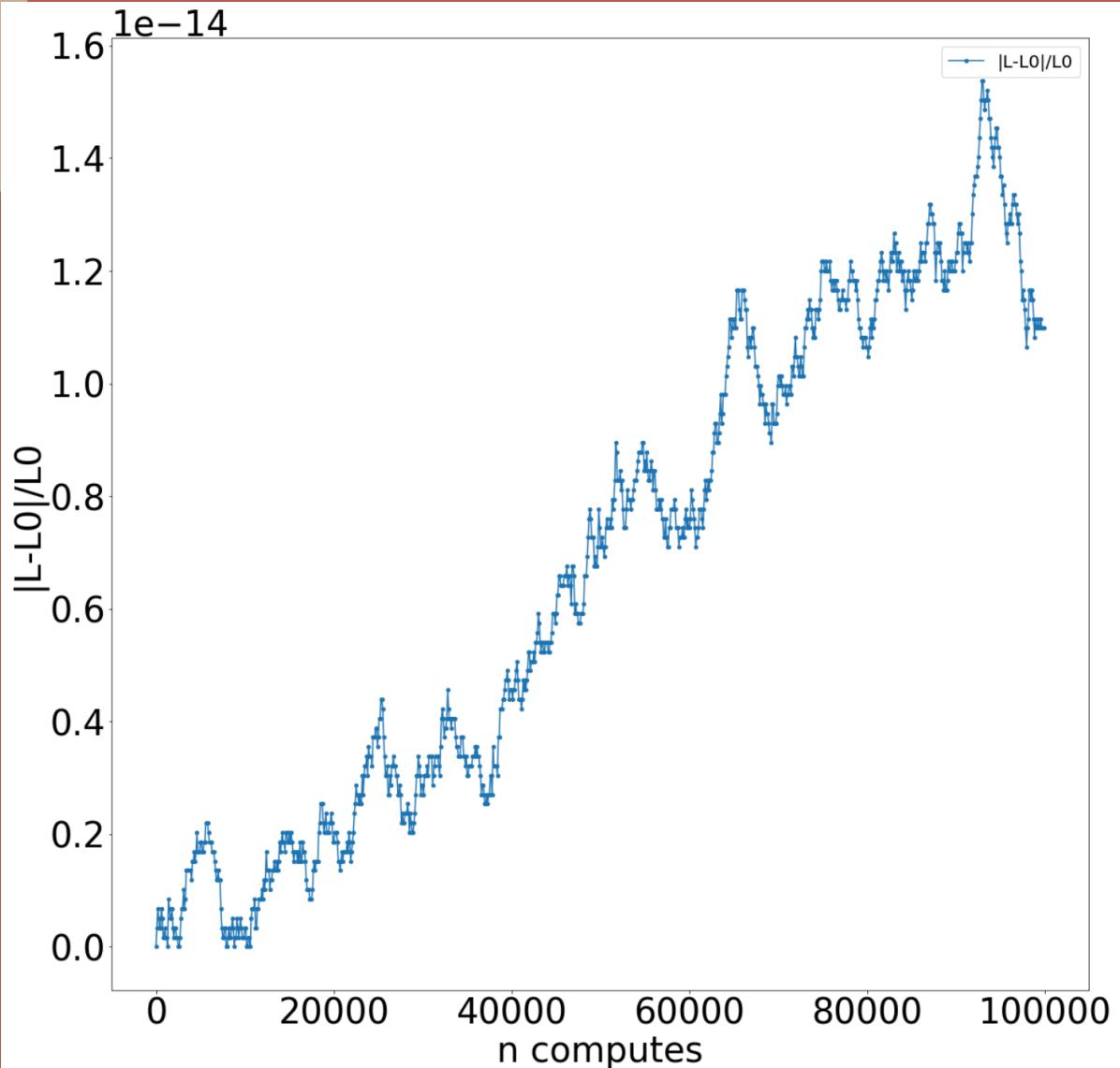
dopri8



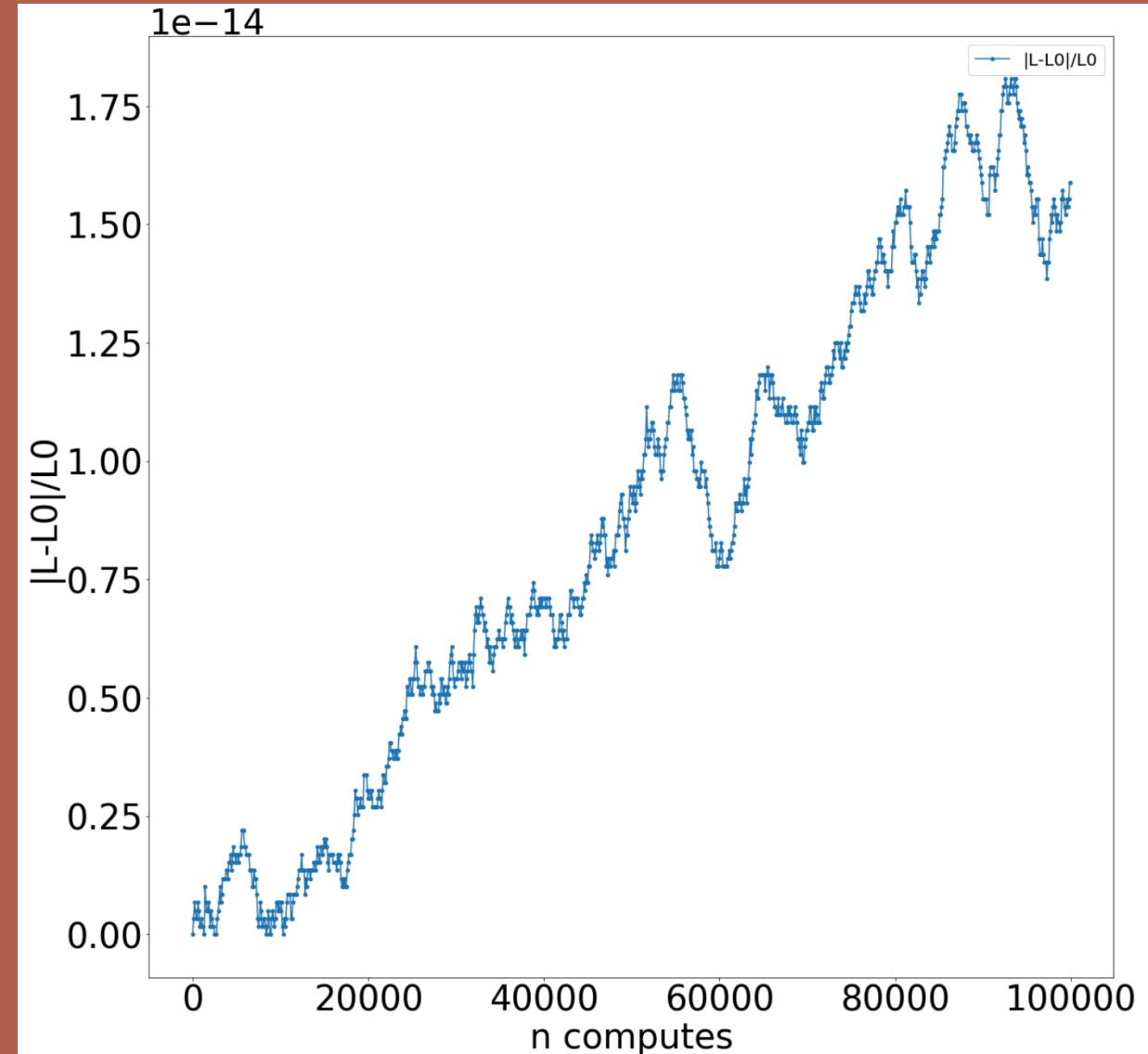
Double(64 bits)

Момент импульса

RK4



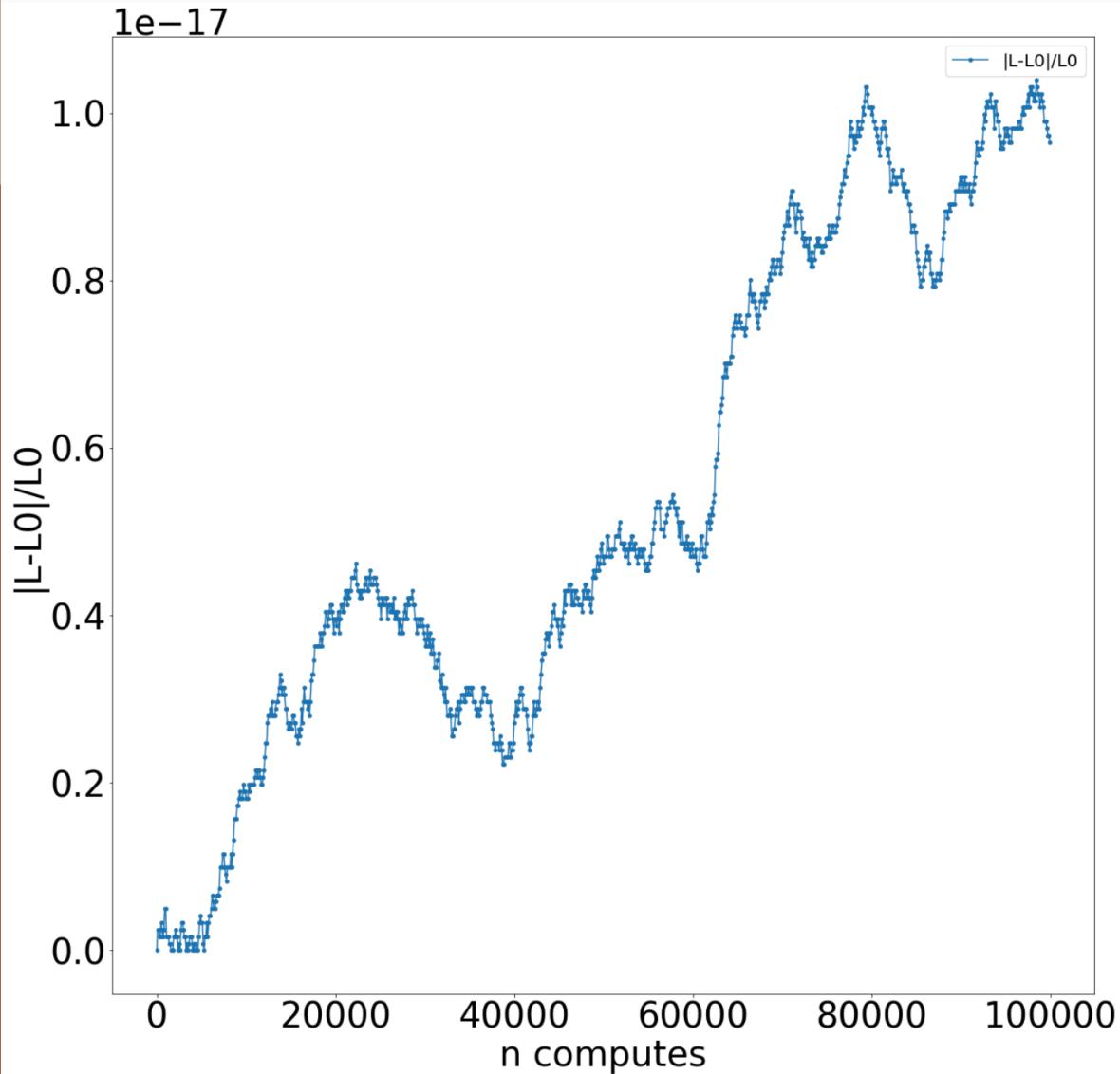
dopri8



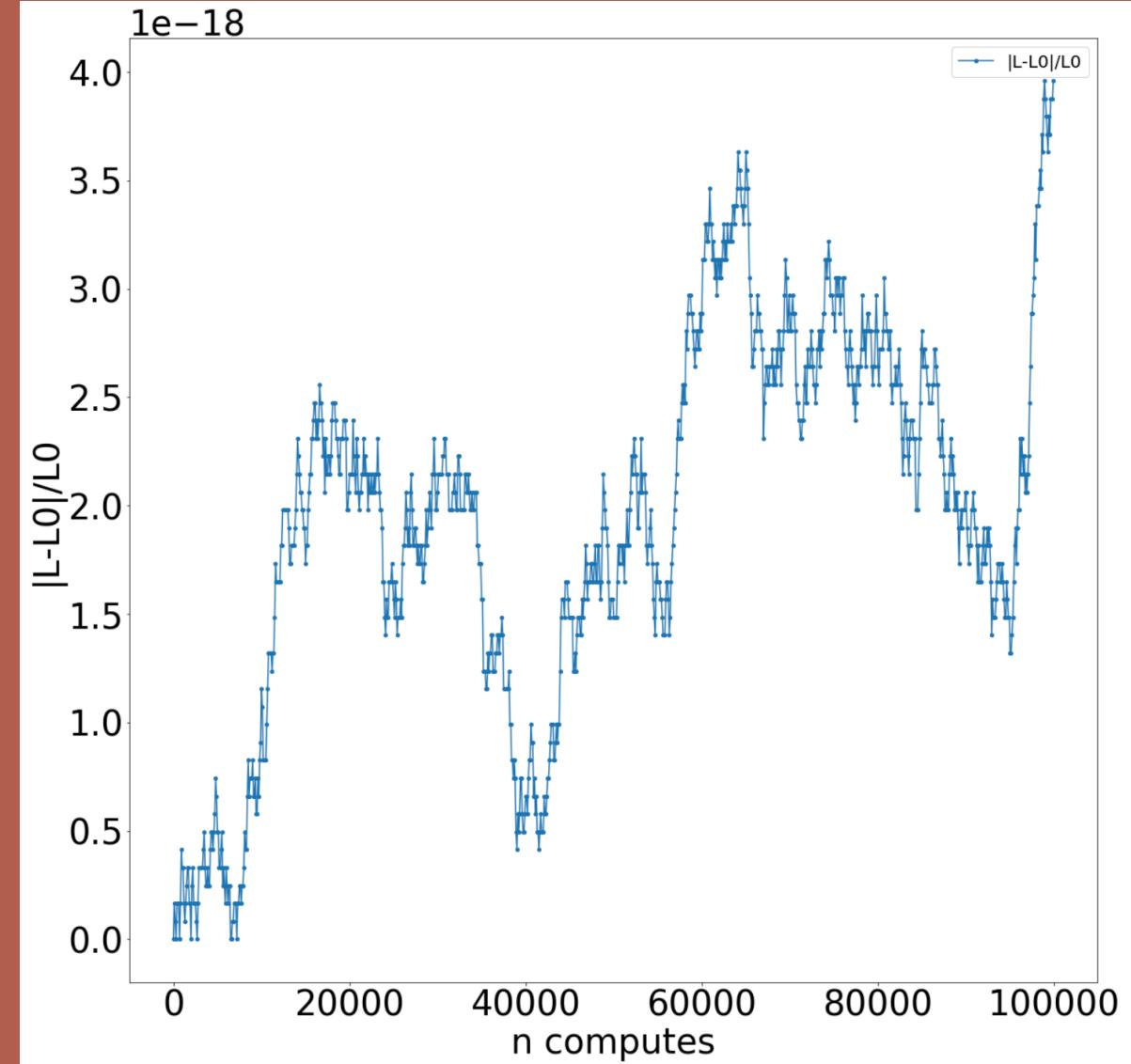
Long double(80 bits)

Момент импульса

RK4



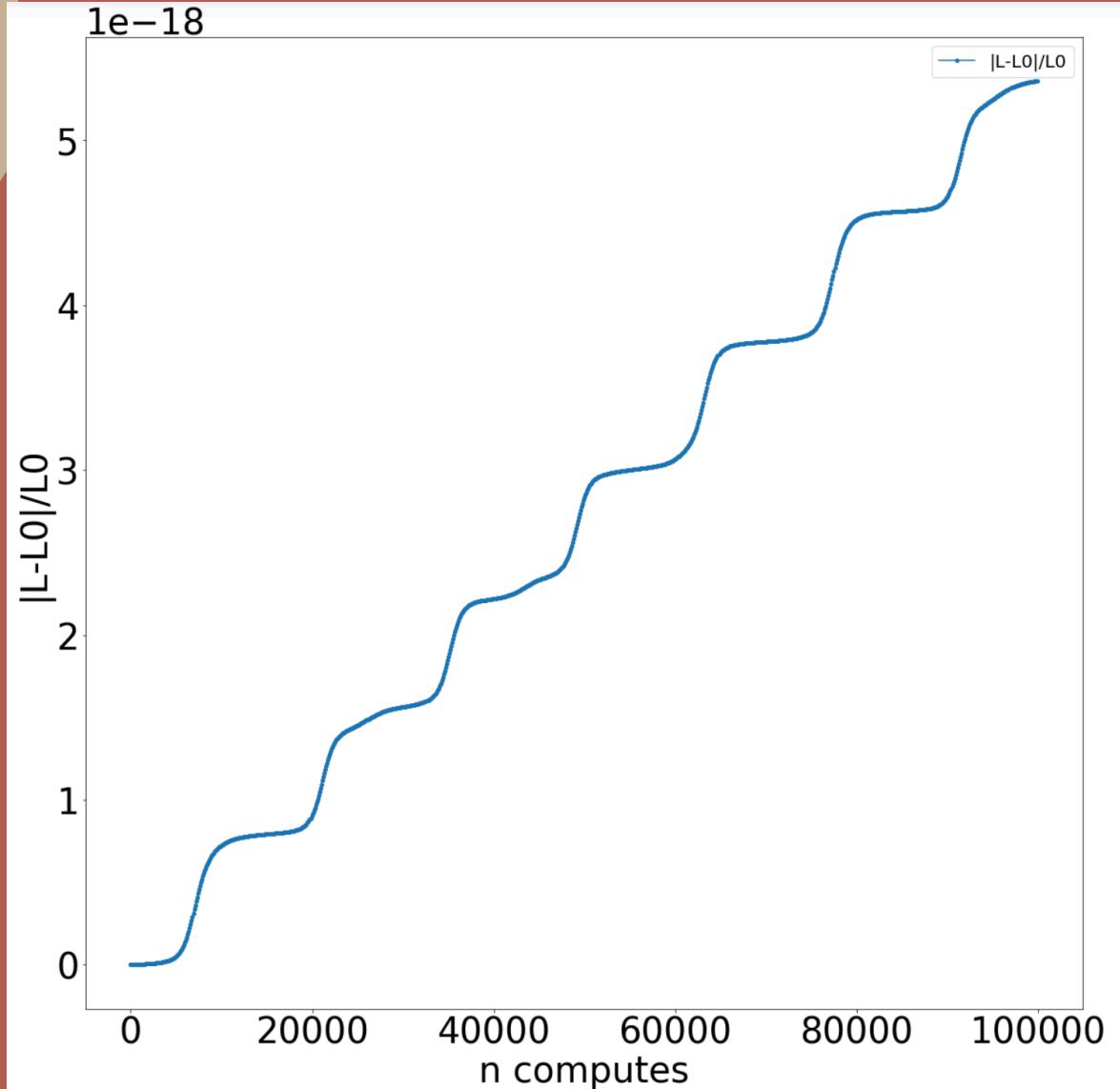
dopri8



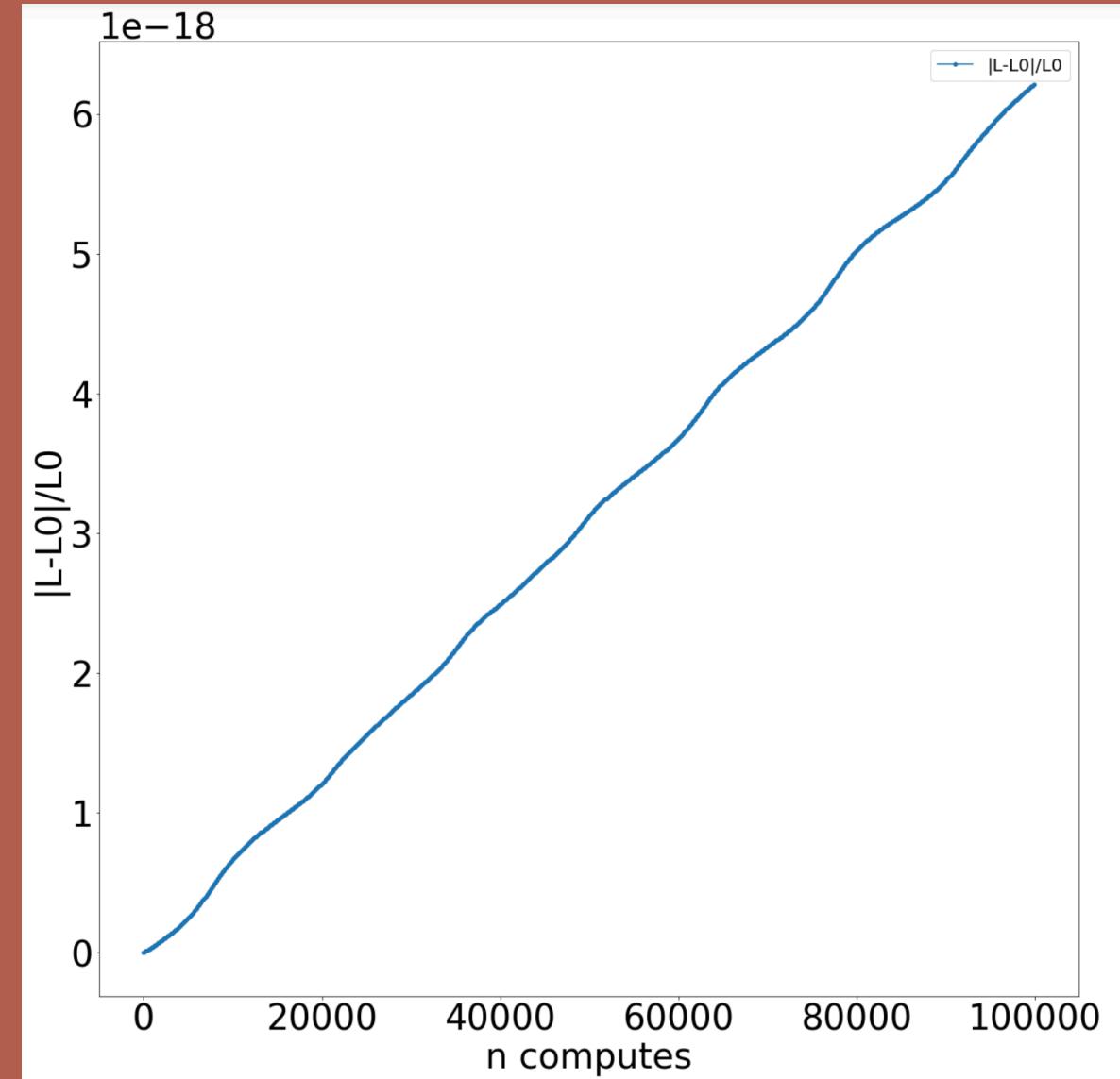
Long double(128 bits)

Момент импульса

RK4

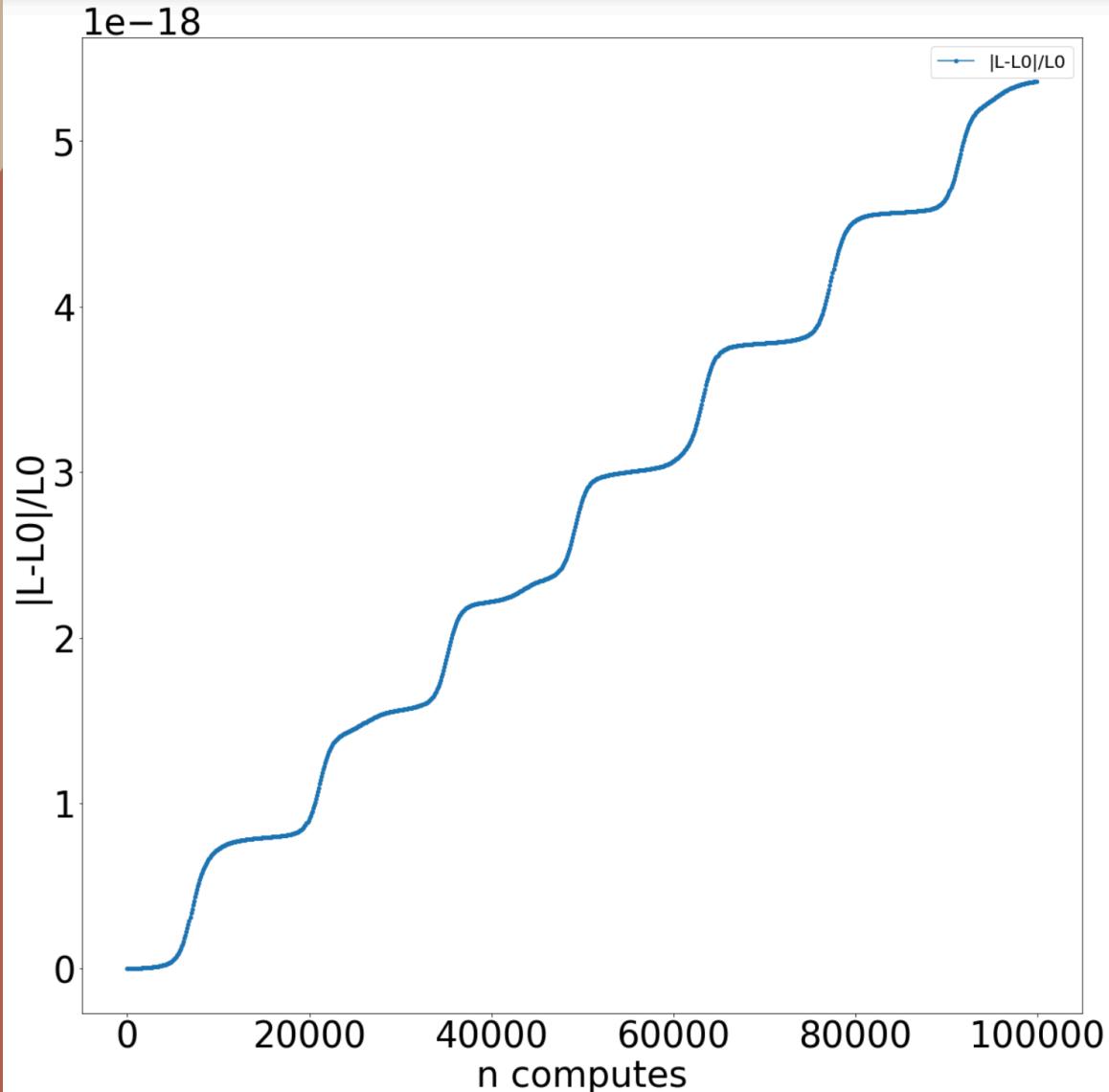


dopri8

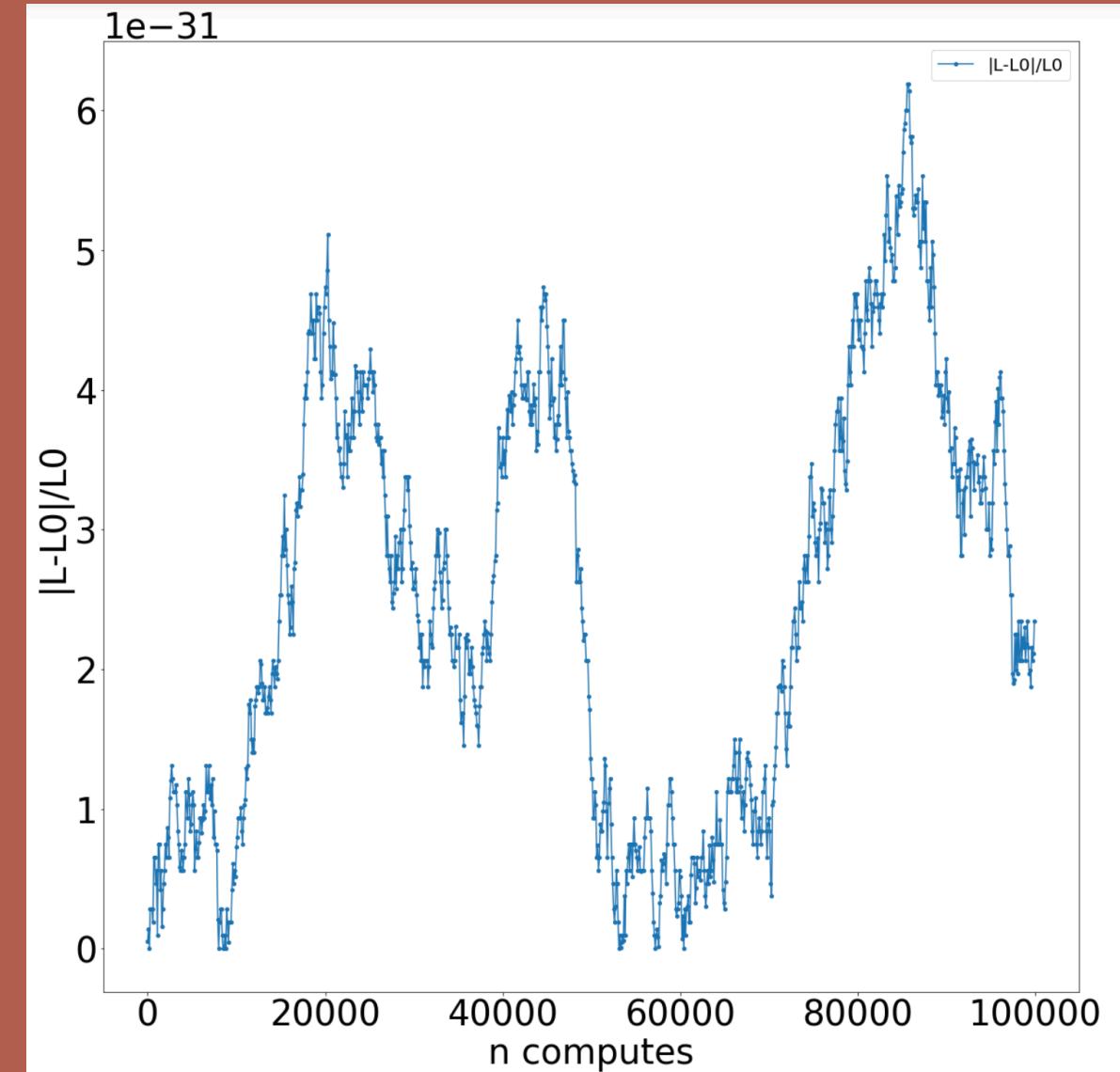


Double-double  
Момент импульса

RK4



dopri8



# Сравнение времени работы

В секундах

	<b>RK4</b>	<b>Dopri8</b>
Double(64 bits)	0.7542	3.275
Long double(80 bits)	1.266	5.925
Long double(128 bits)	14.14	51.33
Double-double	4.653	21.32