|  |  |
| --- | --- |
| Изображение выглядит как текст, керамические изделия, фарфор  Автоматически созданное описание | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

Факультет «Фундаментальные науки»

Кафедра «Техническая физика»

**Домашняя работа №1**

по курсу «Вычислительная физика»

**Вариант 14**

Выполнила: Коберник Т. Н.

Группа: ФН4-81Б

Преподаватели: Хасаншин Р.Х., Ивлиев П.А.

Москва, 2023 г.

**Содержание**

[1.Постановка задачи 3](#_Toc129717896)

[2. Результаты 4](#_Toc129717897)

[3. Вывод 11](#_Toc129717898)

[4. Приложение 12](#_Toc129717899)

1. **Постановка задачи**

Плоский изотропный дисковый источник -квантов (МэВ) накрыт свинцовым кубом. Геометрические центры основания куба и диска совпадают. Ребро куба , а радиус диска см.

**Задание 1.** Построить «ёжика», каждая игла которого представляет отрезок, соединяющий точку рождения -кванта с точкой его первого взаимодействия.

**Задание 2.** Методом локальной оценки потока вычислить распределение плотности потока рассеянных квантов вдоль оси симметрии задачи и вдоль указанных (выделенных на схеме жирными линиями) рёбер куба.

Изображение выглядит как текст, белый

Автоматически созданное описание

**2. Результаты**

Рассмотрим результаты работы программы.

Часть 1

Распределение точек рождения N частиц на поверхности источника.

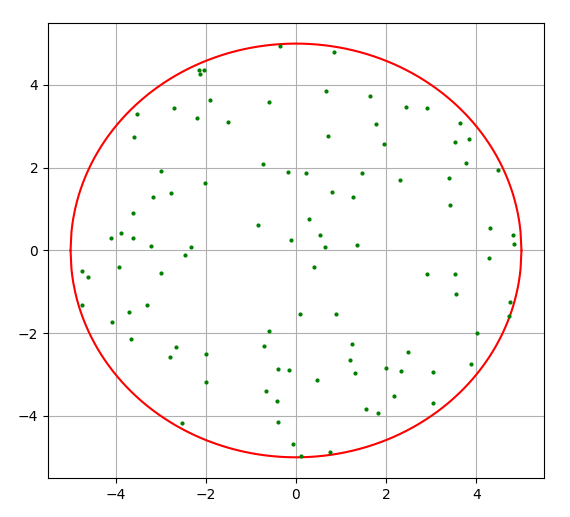
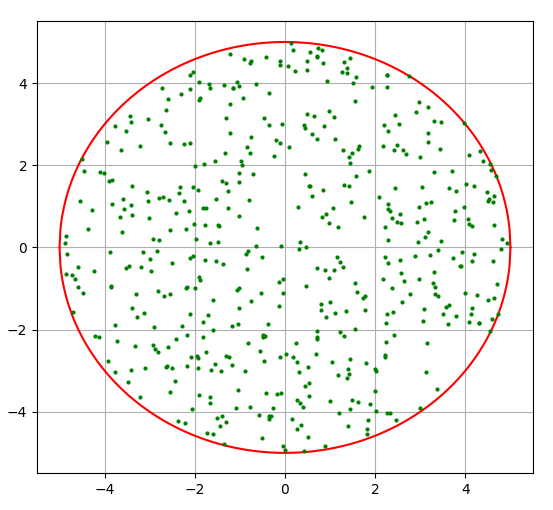
 

Рис. 1а. Точки рождения для N=100 Рис. 1б. Точки рождения для N=500

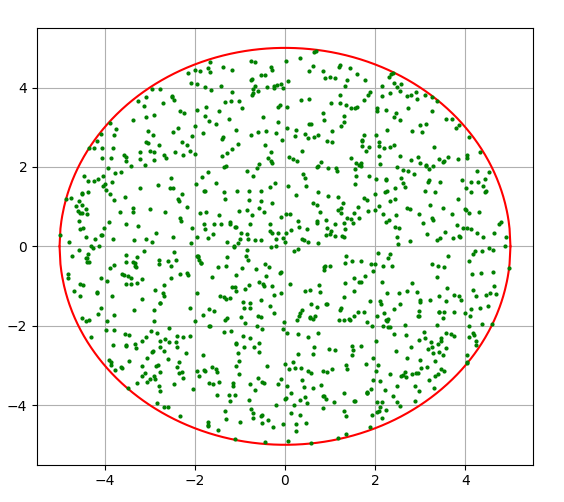
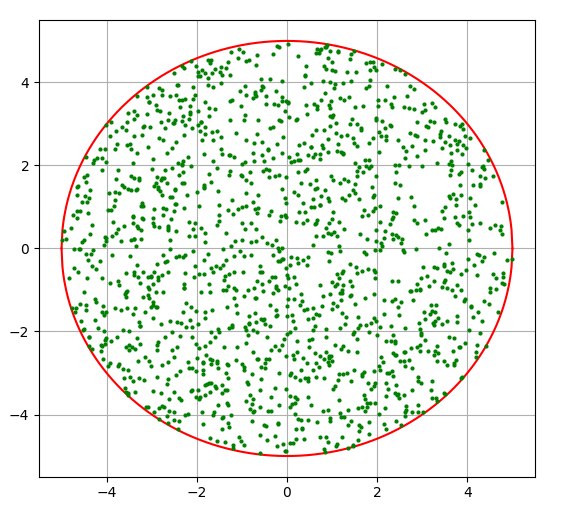
 

Рис. 1в. Точки рождения для N=1000 Рис. 1с. Точки рождения для N=1500

Часть 2

Построение «ёжика» от точек рождения до точек первого взаимодействия

Изображение выглядит как клетка, здание

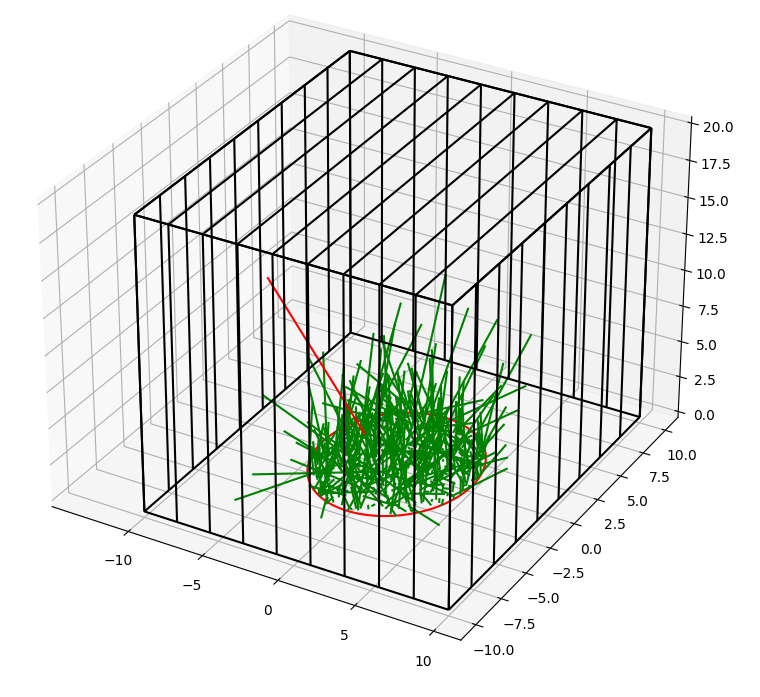
Автоматически созданное описание 

Рис. 2а. «Ёжик» для N=100 Рис. 2б. «Ёжик» для N=500

Изображение выглядит как здание, клетка

Автоматически созданное описаниеИзображение выглядит как здание, клетка

Автоматически созданное описание

Рис. 2в. «Ёжик» для N=1000 Рис. 2с. «Ёжик» для N=1500

Часть 3

Построение полных траекторий движения частиц (от момента рождения до момента исчезновения)

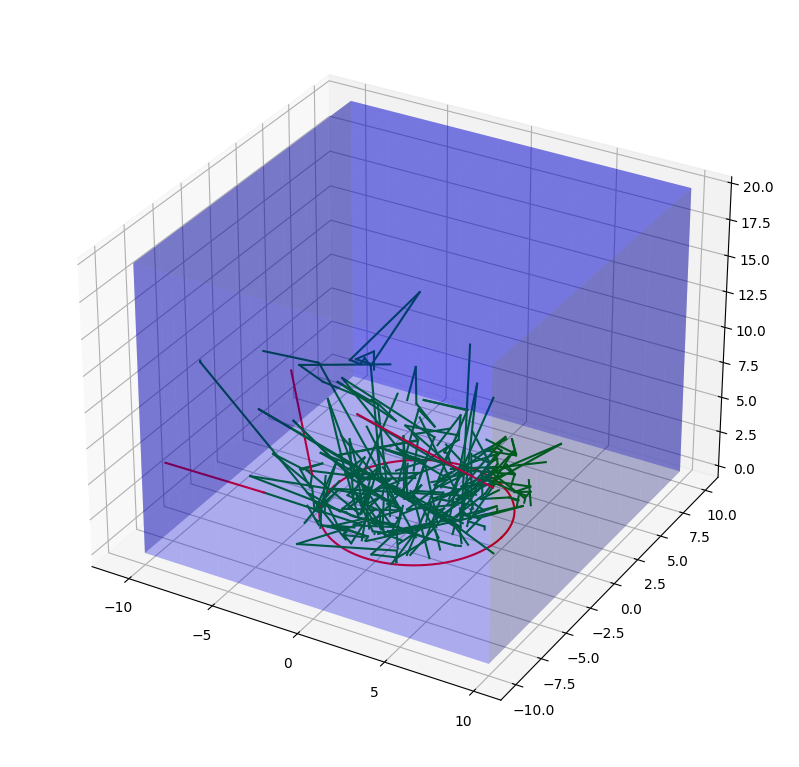
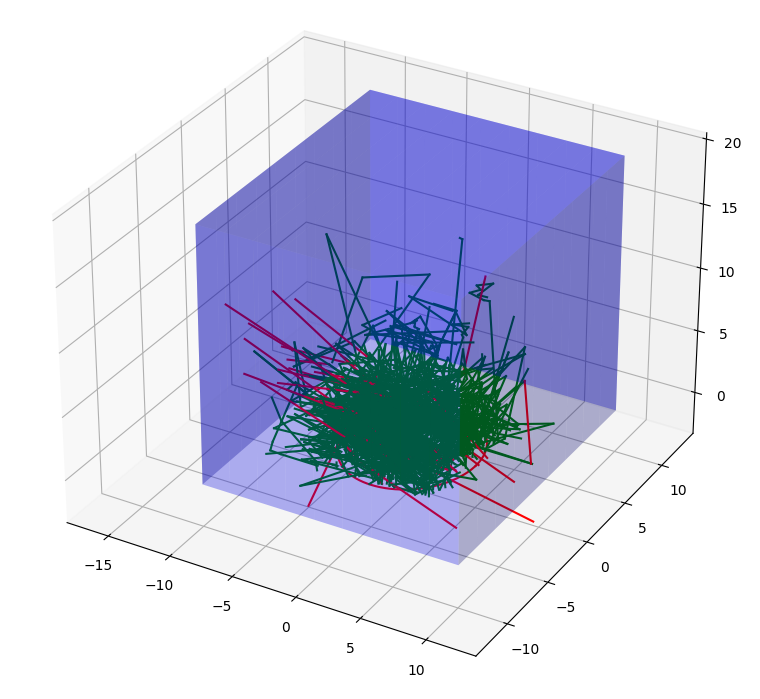
 

Рис. 3а. Полные траектории для N=100 Рис. 3б. Полные траектории для N=500

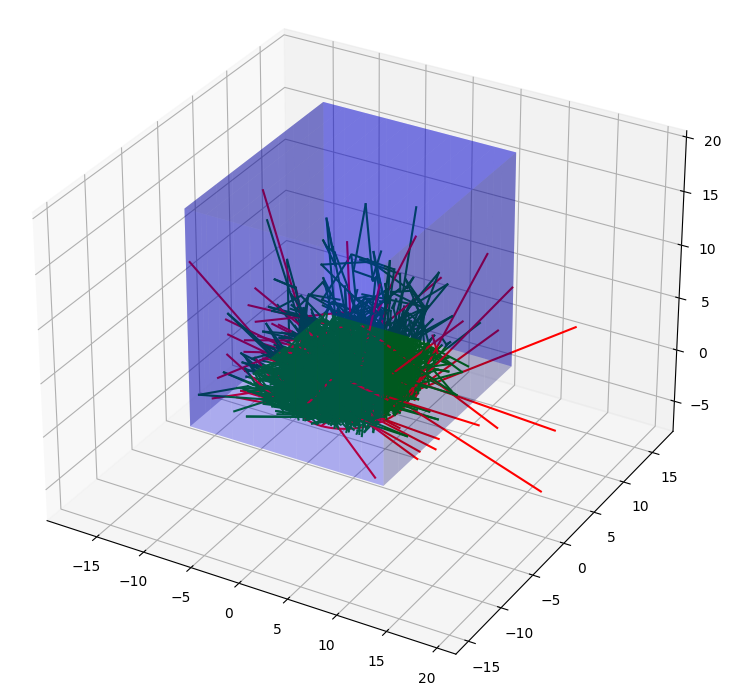
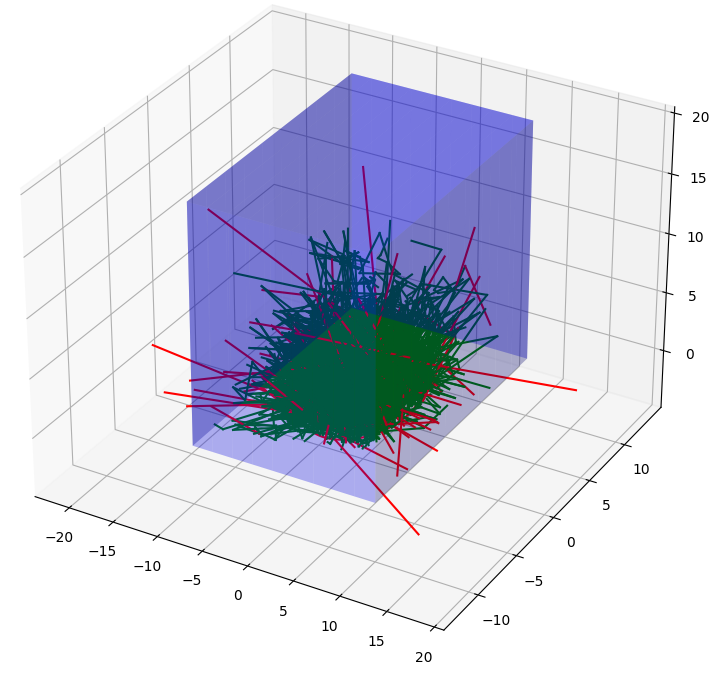
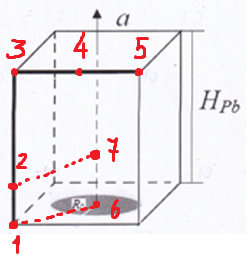
 

Рис. 3в. Полные траектории для N=1000 Рис. 3г. Полные траектории для N=1500

Часть 4

Распределение плотность потока рассеянных квантов было высчитано для 7 точечных детекторов

Расположение датчиков:

Sensor[0] **=** [10, **-**10, 0]

Sensor[1] **=** [10, **-**10, 5]

Sensor[2] **=** [10, **-**10, 20]

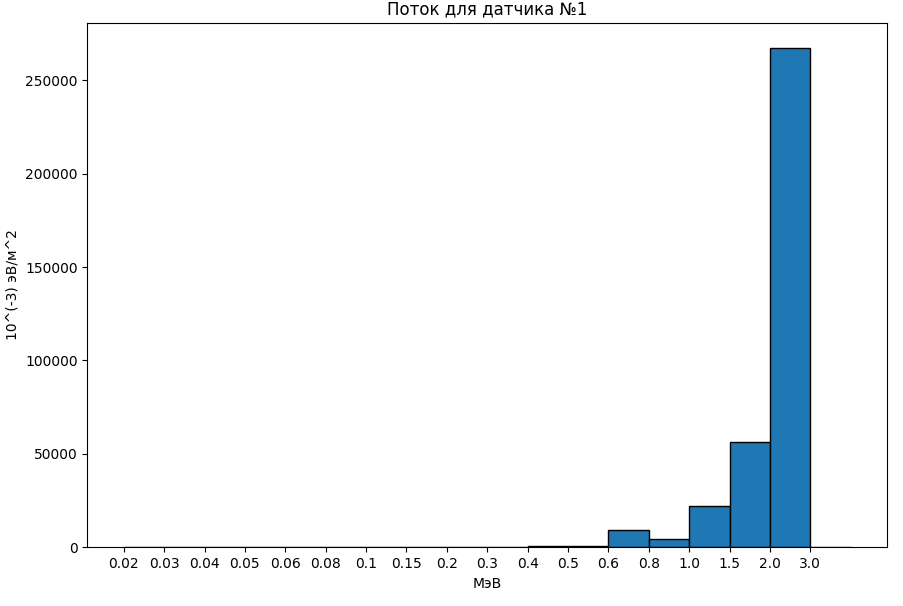
Sensor[3] **=** [10, 0, 20]

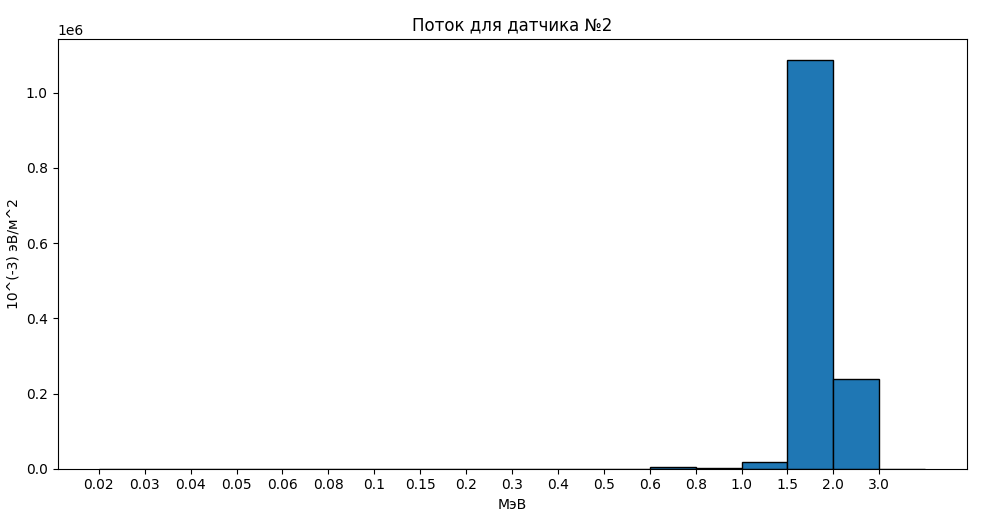
Sensor[4] **=** [10, 10, 20]

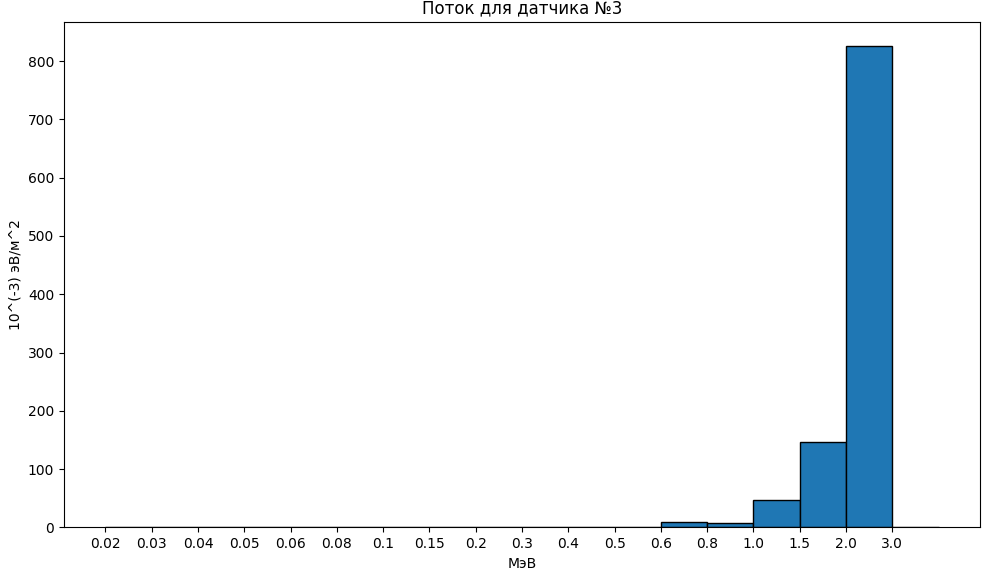
Sensor[5] **=** [0, 0, 0]

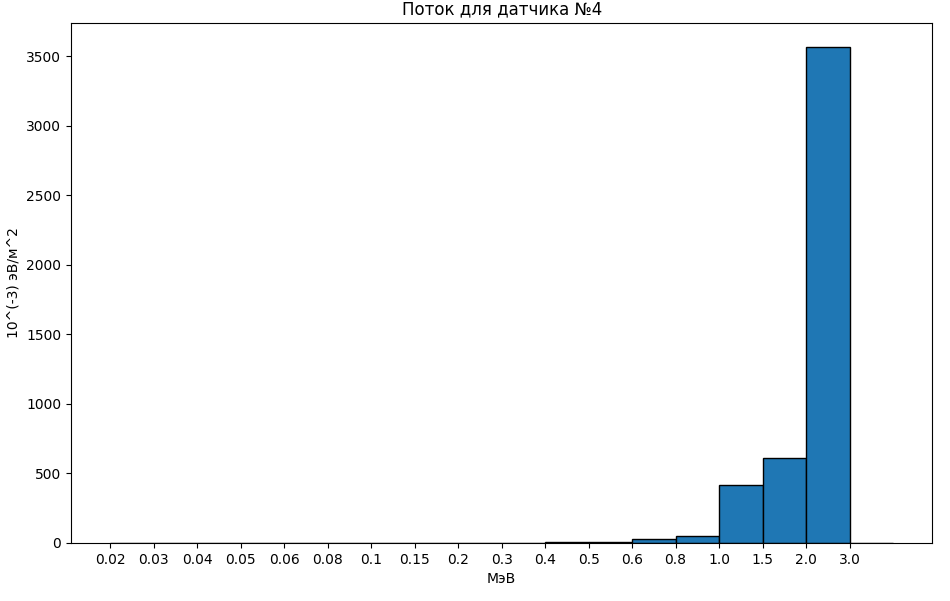
Sensor[6] **=** [0, 0, 5]

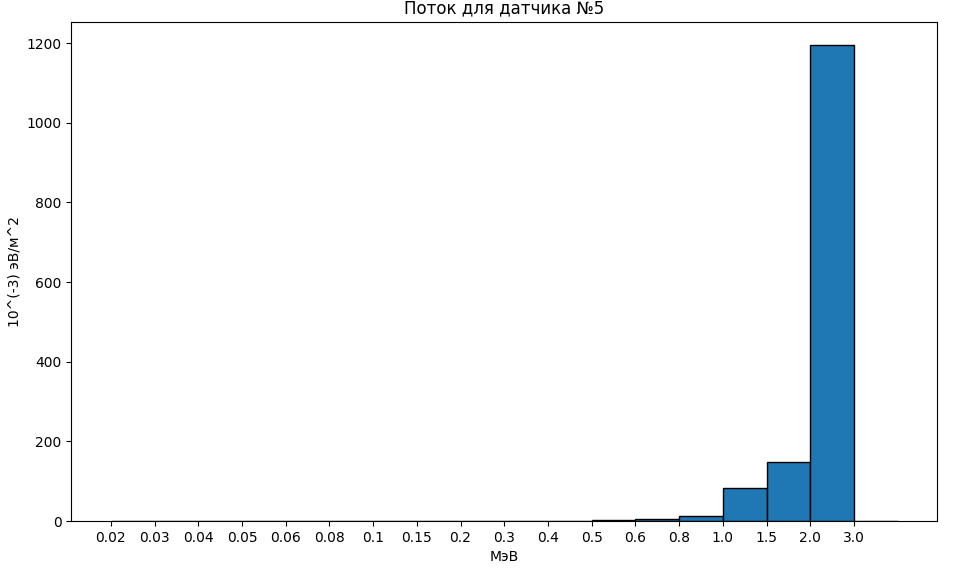
Графики распределения плотности потока для 10000 частиц:

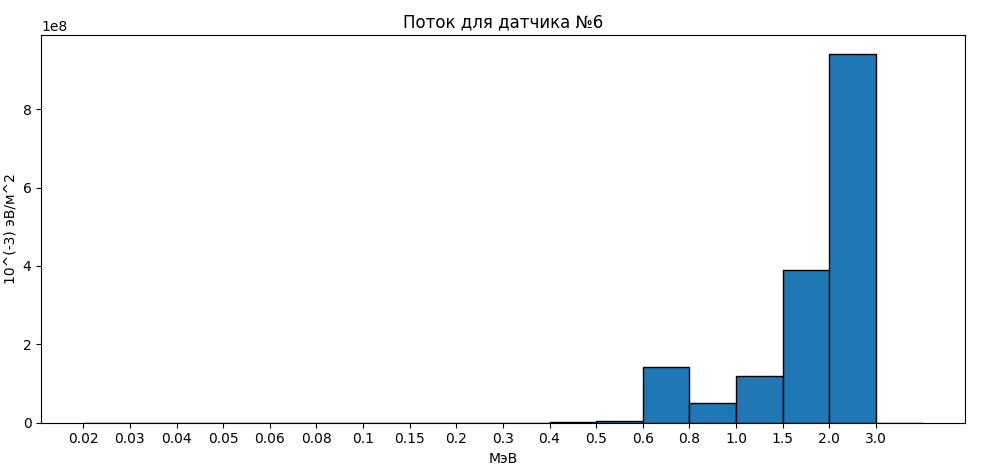


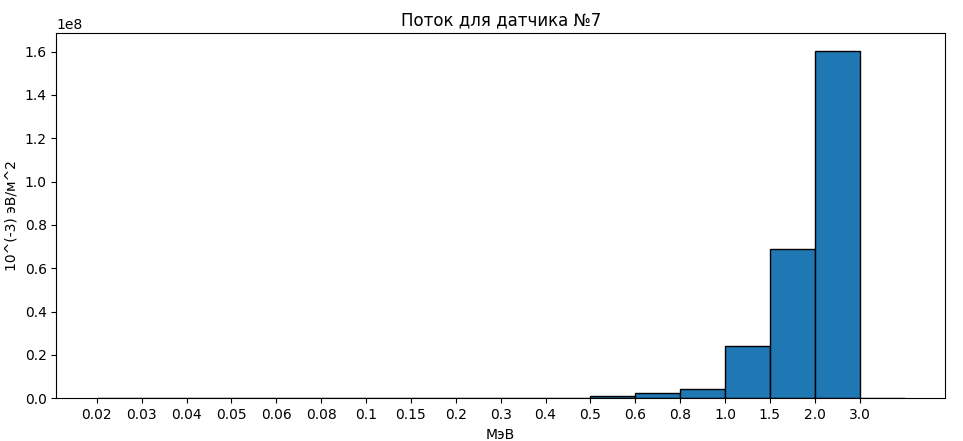












Часть 5

Статистика для различного количества частиц.

*Родилось - 10000 частиц*

Вылетело за пределы области - 89 частиц ( 5.93 % )

Потеряло слишком много энергии - 0 частиц ( 0.0 % )

Недостаточный стат. вес - 99 частиц ( 6.6 % )

Исчезло из-за фотоэффекта - 534 частиц ( 35.6 % )

Исчезло из-за образования эл.-поз. Пары - 778 частиц ( 51.87 % )

В среднем частица переживает 3.63 столкновений

*Родилось - 1500 частиц*

Вылетело за пределы области - 89 частиц ( 5.93 % )

Потеряло слишком много энергии - 0 частиц ( 0.0 % )

Недостаточный стат. вес - 99 частиц ( 6.6 % )

Исчезло из-за фотоэффекта - 534 частиц ( 35.6 % )

Исчезло из-за образования эл.-поз. Пары - 778 частиц ( 51.87 % )

В среднем частица переживает 3.63 столкновений

*Родилось - 1000 частиц*

Вылетело за пределы области - 585 частиц ( 5.85% )

Потеряли слишком много энергии - 0 частиц ( 0.0 % )

Недостаточный стат. вес - 626 частиц ( 6.26 % )

Исчезло из-за фотоэффекта - 3499 частиц ( 34.99 % )

Исчезло из-за образования эл.-поз. пары - 5290 частиц ( 52.9 % )

В среднем частица переживает 3.52 столкновений

*Родилось - 500 частиц*

Вылетело за пределы области - 39 частиц ( 7.8 % )

Потеряли слишком много энергии - 0 частиц ( 0.0 % )

Недостаточный стат. вес - 38 частиц ( 7.6 % )

Исчезло из-за фотоэффекта - 191 частиц ( 38.2 % )

Исчезло из-за образования эл.-поз. пары - 232 частиц ( 46.4 % )

В среднем частица переживает 3.8 столкновений

*Родилось - 100 частиц*

Вылетело за пределы области - 7 частиц ( 7.0 % )

Потеряли слишком много энергии - 0 частиц ( 0.0 % )

Недостаточный стат. вес - 5 частиц ( 5.0 % )

Исчезло из-за фотоэффекта - 30 частиц ( 30.0 % )

из-за образования эл.-поз. пары - 58 частиц ( 58.0 % )

В среднем частица переживает 3.21 столкновений

**3. Вывод**

В данной работе проведено моделирование распространения излучения от плоского источника, накрытого свинцовым цилиндром.

• Получены распределения 100, 500, 1000 и 1500 частиц по поверхности источника;

• Построены траектории 100, 500, 1000 и 1500 частиц от точек рождения до первого взаимодействия;

• Проведены полные расчеты для 10000 частиц спектров плотности потока частиц на заданных детекторах.

После анализа полученных данных можно сделать следующие выводы:

1. В основном частицы поглощались средой за счет образования электрон-позитронных пар.
2. Значение плотности потока частиц максимально на оси симметрии рассмотренной области. На горизонтальном ребре куба плотность потока крайне мала. При чем, стоит отметить, что она почти полностью складывается из плотности потока, созданной высокоэнергичными частицами. Из чего следует, что данное ребро трудно достижимо для частиц.
3. Основной вклад в значение плотности потока частиц внесли частицы с энергиями от 1 до 3 МэВ. Что близко к начальной энергии частиц, испускаемых источником.
4. В ходе моделирования энергия частиц не падала меньше предельно допустимого уровня вследствие комптоновского рассеяния, так как изначальная энергия гамма-квантов, испускаемых источником, крайне велика. Среднее количество столкновений, которое переживают частица составило около 3,5, чего недостаточно для полной потери энергии частицами.

**4. Приложение**

Код использованной программы:

1. **import** math
2. **import** numpy as np
3. **import** random
4. **import** matplotlib.pyplot as plt
6. **def** disk(x, a):
7. **return** (a **\*\*** 2 **-** x **\*\*** 2) **\*\*** (1 **/** 2)
9. **def** p\_x(x, a):
10. p **=** (x **/** a) **+** (a **/** x) **+** ((1 **/** a) **-** (1 **/** x)) **\*** (2 **+** (1 **/** a) **-** (1 **/** x))
11. **return** p
13. **def** sigma\_E\_calk(a, h):
14. A **=** Z\_num **\*** (r\_electron **\*\*** 2)
15. B **=** 1 **-** h
16. C **=** 1 **+** a
17. D **=** 2 **\*** a **+** 1
18. s\_1 **=** A **/** 2 **\*** ((1 **+** a **\*** B) **\*\*** (**-**2)) **\*** (1 **+** (h **\*\*** 2) **+** ((a **\*\*** 2) **\*** (B **\*\*** 2)) **/** (1 **+** a **\*** B))
19. s\_2 **=** 2 **\*** math.pi **\*** A **\*** (
20. (C **/** (a **\*\*** 2)) **\*** ((2 **\*** C **/** D) **-** (math.log(D) **/** a)) **+** (math.log(D) **/** (2 **\*** a)) **-** ((1 **+** 3 **\*** a) **/** (D **\*\*** 2)))
21. **return** [s\_1, s\_2]
23. **def** random\_points\_3(R):
24. gamma\_1 **=** random.uniform(0, 1)
25. gamma\_2 **=** random.uniform(0, 1)
26. r **=** R **\*** (gamma\_1 **\*\*** (1 **/** 2))
27. psi **=** 2 **\*** math.pi **\*** gamma\_2
28. **return** [r **\*** math.cos(psi), r **\*** math.sin(psi)]
30. **def** random\_angles():
31. gamma\_3 **=** random.uniform(0, 1)
32. cos\_tetta **=** 2 **\*** gamma\_3 **-** 1
33. d **=** 2
34. **while** (d > 1):
35. gamma\_4 **=** random.uniform(0, 1)
36. gamma\_5 **=** random.uniform(0, 1)
37. a **=** 2 **\*** gamma\_4 **-** 1
38. b **=** 2 **\*** gamma\_5 **-** 1
39. d **=** ((a **\*\*** 2) **+** (b **\*\*** 2))
40. cos\_phi **=** a **/** (d **\*\*** (1 **/** 2))
41. sin\_phi **=** b **/** (d **\*\*** (1 **/** 2))
42. **return** [sin\_phi, cos\_phi, cos\_tetta]
44. **def** sigma\_count(E):
45. energy\_array **=** [0.02, 0.03, 0.04, 0.05, 0.06, 0.08, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.8, 1.0, 1.5, 2.0, 3.0]
46. sigma **=** [**-**1, **-**1, **-**1]  # вышел за пределы энергий
48. **if** ((E >**=** energy\_array[0]) & (E <**=** energy\_array[17])):
49. **for** i **in** range(len(energy\_array)):
50. buf **=** energy\_array[i]
51. **if** (E **==** buf):
52. sigma[0] **=** SIGMA[i]
53. sigma[1] **=** sig\_comp[i]
54. sigma[2] **=** sig\_ph[i]
55. **elif** (E < buf):
56. sigma[0] **=** SIGMA[i **-** 1] **+** (E **-** energy\_array[i **-** 1]) **\*** (SIGMA[i] **-** SIGMA[i **-** 1]) **/** (
57. buf **-** energy\_array[i **-** 1])
58. sigma[1] **=** sig\_comp[i **-** 1] **+** (E **-** energy\_array[i **-** 1]) **\*** (sig\_comp[i] **-** sig\_comp[i **-** 1]) **/** (
59. buf **-** energy\_array[i **-** 1])
60. sigma[2] **=** sig\_ph[i **-** 1] **+** (E **-** energy\_array[i **-** 1]) **\*** (sig\_ph[i] **-** sig\_ph[i **-** 1]) **/** (
61. buf **-** energy\_array[i **-** 1])
62. **return** sigma
64. **def** run\_lenght(sigma):
65. gamma\_6 **=** random.uniform(0, 1)
66. L **=** **-**math.log(gamma\_6) **/** sigma
67. **return** L
69. **def** check\_position(r, l):
70. **if** ((**-**l **/** 2 <**=** r[0] <**=** l **/** 2) & (**-**l **/** 2 <**=** r[1] <**=** l **/** 2) & (0 <**=** r[2] <**=** l)):
71. **return** 1
72. **else**:
73. **return** 0
75. **def** interaction\_type(s):
76. sum **=** s[0] **/** P
77. gamma\_7 **=** random.uniform(0, 1)
78. **if** (gamma\_7 < (s[1] **/** sum)):
79. **return** 1
80. **elif** (gamma\_7 < ((s[1] **+** s[2]) **/** sum)):
81. **return** 2
82. **else**:
83. **return** 0
85. **def** random\_energy(a):
86. p **=** 1
87. coef **=** 0
88. x **=** 0
89. **while** (p >**=** coef):
90. gamma\_1 **=** random.uniform(0, 1)
91. gamma\_2 **=** random.uniform(0, 1)
93. x **=** (a **\*** (1 **+** 2 **\*** a **\*** gamma\_1)) **/** (1 **+** 2 **\*** a)
94. p **=** p\_x(x, a)
95. coef **=** gamma\_2 **\*** (1 **+** 2 **\*** a **+** (1 **/** (1 **+** 2 **\*** a)))
96. **return** x
98. **def** energy\_group(E):
99. K **=** 0
100. **if** 0.02 <**=** E <**=** 0.03:
101. K **=** 1
102. **elif** 0.03 < E <**=** 0.04:
103. K **=** 2
104. **elif** 0.04 < E <**=** 0.05:
105. K **=** 3
106. **elif** 0.05 < E <**=** 0.06:
107. K **=** 4
108. **elif** 0.06 < E <**=** 0.08:
109. K **=** 5
110. **elif** 0.08 < E <**=** 0.1:
111. K **=** 6
112. **elif** 0.1 < E <**=** 0.15:
113. K **=** 7
114. **elif** 0.15 < E <**=** 0.2:
115. K **=** 8
116. **elif** 0.2 < E <**=** 0.3:
117. K **=** 9
118. **elif** 0.3 < E <**=** 0.4:
119. K **=** 10
120. **elif** 0.4 < E <**=** 0.5:
121. K **=** 11
122. **elif** 0.5 < E <**=** 0.6:
123. K **=** 12
124. **elif** 0.6 < E <**=** 0.8:
125. K **=** 13
126. **elif** 0.8 < E <**=** 1.0:
127. K **=** 14
128. **elif** 1.0 < E <**=** 1.5:
129. K **=** 15
130. **elif** 1.5 < E <**=** 2.0:
131. K **=** 16
132. **elif** 2.0 < E <**=** 3.0:
133. K **=** 17
134. **return** (K **-** 1)
136. **def** W\_count(angle):
137. W **=** [0] **\*** 3
138. W[0] **=** angle[1] **\*** angle[2]
139. W[1] **=** angle[0] **\*** angle[2]
140. W[2] **=** pow((1 **-** pow(angle[2], 2)), 1 **/** 2)
141. **return** W
143. **def** W\_recount(W, angle):
144. W\_res **=** [0] **\*** 3
146. q **=** ((1 **-** (angle[2] **\*\*** 2)) **\*** (1 **-** (W[2] **\*\*** 2))) **\*\*** (1 **/** 2)
147. W\_res[2] **=** W[2] **\*** angle[2] **+** q **\*** angle[1]
149. a **=** angle[2] **-** W[2] **\*** W\_res[2]
150. b **=** angle[0] **\*** q
151. c **=** 1 **-** (W[2]) **\*\*** 2
153. W\_res[1] **=** ((W[1] **\*** a) **+** (W[0] **\*** b)) **/** c
154. W\_res[0] **=** ((W[0] **\*** a) **-** (W[1] **\*** b)) **/** c
155. **return** W\_res
157. **def** eta\_calc(r, cos\_tetta, a\_new, W, Sigma):
158. eta **=** [0, 0, 0, 0, 0, 0, 0]
159. Sensor **=** [[0] **\*** 3] **\*** 7
160. Sensor[0] **=** [10, **-**10, 0]
161. Sensor[1] **=** [10, **-**10, 5]
162. Sensor[2] **=** [10, **-**10, 20]
163. Sensor[3] **=** [10, 0, 20]
164. Sensor[4] **=** [10, 10, 20]
165. Sensor[5] **=** [0, 0, 0]
166. Sensor[6] **=** [0, 0, 5]
168. sigma\_E **=** sigma\_E\_calk(a\_new, cos\_tetta)
169. scat\_indicatrix **=** W **\*** sigma\_E[0] **/** sigma\_E[1]
171. **for** i **in** range(len(eta)):
172. delta\_r **=** ((r[0] **-** Sensor[i][0]) **\*\*** 2) **+** ((r[1] **-** Sensor[i][1]) **\*\*** 2) **+** ((r[2] **-** Sensor[i][2]) **\*\*** 2)
173. eta[i] **=** scat\_indicatrix **\*** math.exp(**-** Sigma **\*** (delta\_r **\*\*** (1 **/** 2))) **/** delta\_r
175. **return** eta
177. sig\_comp **=** (0.147, 0.142, 0.138, 0.134, 0.130, 0.123,
178. 0.117, 0.106, 0.0968, 0.0843, 0.0756, 0.0689,
179. 0.0637, 0.0561, 0.0503, 0.0410, 0.0349, 0.0274)
180. sig\_ph **=** (83.1, 28.5, 13.2, 7.21, 4.39,
181. 1.97, 5.23, 1.8, 0.843, 0.289,
182. 0.141, 0.0823, 0.0538, 0.0285, 0.0180, 0.00858, 0.00523, 0.00282)
183. sig\_pair **=** (0, 0, 0, 0, 0, 0, 0, 0,
184. 0, 0, 0, 0, 0,
185. 0, 0, 0.00159, 0.005, 0.0115)
187. SIGMA **=** [0] **\*** len(sig\_comp)
189. P **=** 11.35  # плотность свинца
190. Z\_num **=** 82  # атомный номер свинца
191. r\_electron **=** 2.8 **\*** (10 **\*\*** (**-**13))  # радиус электрона
193. **for** i **in** range(len(SIGMA)):
194. SIGMA[i] **=** (sig\_comp[i] **+** sig\_ph[i] **+** sig\_pair[i]) **\*** P
196. x **=** np.linspace(**-**5, 5, 500)
197. plt.grid(True)
198. plt.figure(1)
199. plt.plot(x, disk(x, 5), color**=**'r')
200. plt.plot(x, **-**disk(x, 5), color**=**'r')
202. **for** k **in** range(500):
203. point **=** random\_points\_3(5)
204. plt.plot(point[0], point[1], 'go', markersize**=**2)  # график разброса точек на плоскости
206. fig **=** plt.figure(2)
207. ax **=** fig.add\_subplot(111, projection**=**'3d')
208. ax.plot(x, disk(x, 5), color**=**'r')
209. ax.plot(x, **-**disk(x, 5), color**=**'r')
211. x **=** np.linspace(**-**10, 10, 10)
212. y **=** np.linspace(**-**10, 10, 10)
213. z **=** np.linspace(0, 20, 10)
215. values **=** [20, 10, **-**10, 10, **-**10]
217. **for** i **in** range(5):
218. **if** (i **==** 0):
219. X, Y **=** np.meshgrid(x, y)
220. eq **=** 20 **\*** (X **-** X) **+** values[i]
221. ax **=** fig.gca(projection**=**'3d')
222. ax.plot\_surface(X, Y, eq, rcount**=**1, color**=**'blue', alpha**=**0.3)
223. **elif** ((i > 0) & (i < 3)):
224. Y, Z **=** np.meshgrid(y, z)
225. eq **=** 20 **\*** (Y **-** Y) **+** values[i]
226. ax **=** fig.gca(projection**=**'3d')
227. ax.plot\_surface(eq, Y, Z, rcount**=**1, color**=**'blue', alpha**=**0.3)
228. **elif** (i >**=** 3):
229. X, Z **=** np.meshgrid(x, z)
230. eq **=** 20 **\*** (X **-** X) **+** values[i]
231. ax **=** fig.gca(projection**=**'3d')
232. ax.plot\_surface(X, eq, Z, rcount**=**1, color**=**'blue', alpha**=**0.3)
234. # die = 0
235. ph **=** 0
236. pair **=** 0
237. fly **=** 0
238. small\_energy **=** 0
239. small\_weight **=** 0
240. n **=** 10000
241. R **=** [0, 0]
242. counter **=** 0
243. counter\_sum **=** 0
244. PHI **=** np.empty((7, 18))
245. eta **=** np.empty((7, 18))
247. **for** j **in** range(n):
248. **for** l **in** range(7):
249. **for** q **in** range(17):
250. PHI[l][q] **+=** eta[l][q]
251. eta **=** np.empty((7, 18))
253. W\_stat **=** 1  # статистический вес частицы
254. counter\_sum **+=** counter
256. r\_0 **=** [random\_points\_3(5)[0], random\_points\_3(5)[1], 0]
257. R[0] **=** r\_0
258. angle **=** random\_angles()
260. energy\_old **=** 3.0  # н.у.
261. K **=** 17  # номер энергетической группы
262. sigma **=** sigma\_count(energy\_old)
263. L\_1 **=** run\_lenght(sigma[0])
265. W **=** W\_count(angle)
266. r\_1 **=** [r\_0[0] **+** L\_1 **\*** W[0], r\_0[1] **+** L\_1 **\*** W[1], L\_1 **\*** W[2]]
267. R[1] **=** r\_1
269. counter **=** 1
270. kost **=** 1
272. **while** (kost !**=** 0):
273. check **=** check\_position(R[1], 20)
274. **if** (check **==** 1):
275. ax.plot([R[0][0], R[1][0]], [R[0][1], R[1][1]], [R[0][2], R[1][2]], 'g-')
276. **else**:
277. fly **+=** 1
278. ax.plot([R[0][0], R[1][0]], [R[0][1], R[1][1]], [R[0][2], R[1][2]], 'r-')
279. kost **=** 0
280. **continue**
281. inter **=** interaction\_type(sigma)
282. **if** inter **==** 0:
283. pair **+=** 1
284. kost **=** 0
285. **continue**
286. **if** inter **==** 2:
287. ph **+=** 1
288. kost **=** 0
289. **continue**
291. energy\_new **=** random\_energy(energy\_old)
292. **if** (energy\_new <**=** 0.02):
293. small\_energy **+=** 1
294. kost **=** 0
295. **continue**
296. K **=** energy\_group(energy\_new)  # номер энергетической группы
298. sigma **=** sigma\_count(energy\_new)  # [0] - SIGMA, [1] - sigma\_comp
299. W\_stat **=** W\_stat **\*** sigma[1] **/** sigma[0]  # статистический вес частицы
300. **if** (W\_stat < (10 **\*\*** (**-**11))):
301. small\_weight **+=** 1
302. kost **=** 0
303. **continue**
304. angle\_scat **=** [random\_angles()[0], random\_angles()[1], (1 **+** (1 **/** energy\_old) **-** (
305. 1 **/** energy\_new
306. # оценка потока
307. eta\_new **=** eta\_calc(R[1], angle\_scat[2], energy\_new, W\_stat, sigma[0])
308. **for** f **in** range(7):
309. eta[f][K] **+=** eta\_new[f]  # записываем отдельно вклад от каждой энергетической группы
310. # дальнейшая траектория частицы
311. W **=** W\_count(angle\_scat)
312. W **=** W\_recount(W, angle\_scat)  # вернулись в изначальную СК
313. L\_2 **=** L\_1 **=** run\_lenght(sigma[0])
314. r\_2 **=** [r\_1[0] **+** L\_2 **\*** W[0], r\_1[1] **+** L\_2 **\*** W[1], r\_1[2] **+** L\_2 **\*** W[2]]
315. R[0] **=** R[1]
316. R[1] **=** r\_2
317. counter **+=** 1
318. **for** k **in** range(7):
319. **for** l **in** range(17):
320. PHI[k][l] **=** round(PHI[k][l] **/** n **\*** (10 **\*\*** 9), 1)
322. print('\nВас приветствует Hedgehog (версия 2.1)\nСтатистика на сегодня:')
323. print("\nРодилось - ", n, "частиц")
324. print("Вылетело за пределы области - ", fly, "частиц (", round(fly **/** n **\*** 100, 2), "% )")
325. print("Потеряли слишком много энергии - ", small\_energy, "частиц (", round(small\_energy **/** n **\*** 100, 2), "% )")
326. print("Недостаточный стат. вес - ", small\_weight, "частиц (", round(small\_weight **/** n **\*** 100, 2), "% )")
327. # print("Исчезло из-за взаимодействий- ", die, "частиц (", round(die/n \* 100, 2), "% )\n")
328. print("Исчезло из-за фотоэффекта- ", ph, "частиц (", round(ph **/** n **\*** 100, 2), "% )")
329. print("Исчезло из-за образ.пары- ", pair, "частиц (", round(pair **/** n **\*** 100, 2), "% )\n")
330. print("В среднем частица переживает", round(counter\_sum **/** n, 2), "столкновений")
331. # построение гистограммы
332. labels **=** ['0.02', '0.03', '0.04', '0.05', '0.06', '0.08', '0.1', '0.15', '0.2', '0.3', '0.4', '0.5', '0.6', '0.8',
333. '1.0', '1.5', '2.0', '3.0']
334. x\_f **=** [0.02, 0.03, 0.04, 0.05, 0.06, 0.08, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.8, 1.0, 1.5, 2.0]
335. x\_g **=** [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
336. **for** k **in** range(7):
337. fig **=** plt.figure(2 **+** k)
338. fig\_g, ax **=** plt.subplots(figsize**=**(8,4))
339. ax.bar(x\_g, PHI[k], align**=**'edge', edgecolor**=**'black', width**=**1)
340. ax.set\_xticks(x\_g)
341. ax.set\_xticklabels(labels)
342. fig\_g.tight\_layout()
343. ax.set\_xlabel('МэВ')
344. ax.set\_ylabel('10^(-3) эВ/м^2')
345. ax.set\_title('Поток для датчика №' **+** str(k**+**1))
347. plt.show()