

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ им. ПЕТРА ВЕЛИКОГО

ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ И МЕХАНИКИ

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ

## ЛАБОРАТОРНАЯ РАБОТА №2

4 КУРС, ГРУППА 3630102/60201

Студент

Д. А. Плаксин

Преподаватель

Баженов А. Н.

САНКТ-ПЕТЕРБУРГ  
2019 г.

# Содержание

<b>1. Список иллюстраций .....</b>	<b>3</b>
<b>2. Постановка задачи .....</b>	<b>4</b>
<b>3. Теория.....</b>	<b>4</b>
3.1. Сечение тела вращения .....	4
3.2. Лемниската Бернулли .....	4
<b>4. Реализация.....</b>	<b>4</b>
<b>5. Результаты .....</b>	<b>5</b>
5.1. Построение сечений .....	5
5.2. Сравнение с лемниской Бернулли.....	9
<b>6. Обсуждение.....</b>	<b>10</b>
<b>7. Список литературы .....</b>	<b>10</b>
<b>8. Приложения .....</b>	<b>11</b>

## 1 Список иллюстраций

1	Сечение тора плоскость $x = 0$ .....	5
2	Сечение тора плоскость $x = 5$ .....	5
3	Сечение тора плоскость $x = 10$ .....	6
4	Сечение тора плоскость $x = 15$ .....	6
5	Сечение тора плоскость $x = 20$ .....	7
6	Сечение тора плоскость $x = 25$ .....	7
7	Сечение тора плоскость $x = 35$ .....	8
8	Все сечения на одном графике .....	8
9	Лемниската для сечения тора .....	9
10	Лемниската практически совпадающая с сечением тора .....	10

## 2 Постановка задачи

Построить фигуру вращения – тор, в виде дискретного набора точек в пространстве. [4]

Построить набор сечений фигуры плоскостями  $x = H$ .

Одно из сечений хорошо описывается лемнискатой Бернулли. Необходимо построить лемнискату и сравнить её с соответствующим сечением.

Варьируя параметры лемнискаты и тора минимизировать расстояние (Фреше) между ними.

## 3 Теория

### 3.1 Сечение тела вращения

Сечение задаётся в плоскости  $XOZ$ . Само тело получается путём вращения плоскости сечения с точками вокруг оси  $Z$ . [4]

Так как сечение тела представляется в виде дискретного набора точек, и каждая из точек движется по окружности, то нужно для каждой точки найти пересечение её окружности и плоскости  $x = H$ . Пересечение ищется с помощью системы:

$$\begin{cases} x^2 + y^2 = R^2 \\ x = H \\ z = z' \end{cases} \quad (1)$$

Решение системы:  $(H, \pm\sqrt{R^2 - H^2}, z')$ . Если подкоренное выражение меньше нуля, то пересечения нет.

### 3.2 Лемниската Бернулли

Параметрическое уравнение лемнискаты Бернулли: [4]

$$\begin{cases} y = c \frac{t+t^3}{1+t^4} \\ x = c \frac{t-t^3}{1+t^4} \\ z = z' \end{cases} \quad (2)$$

где  $t = \operatorname{tg}(\varphi)$ ,  $c$  – параметр.

## 4 Реализация

Для реализации лабораторной использовался язык *Python* 3.7. Использовалась библиотека *numpy*. [1]

Графики строились с помощью библиотеки *matplotlib*. [2]

Трёхмерные графики строились с помощью библиотеки *mpl\_toolkits.mplot3d*. [3]

## 5 Результаты

### 5.1 Построение сечений

Рассматривается тор со следующими характеристиками:  $R = 20$  – радиус вращения,  $r = 15$  – радиус образующей.

Рассматриваемы плоскости сечения  $x = H$ , где  $H = \{0, 5, 10, 15, 20, 25, 35\}$ .

Рис. 1: Сечение тора плоскость  $x = 0$

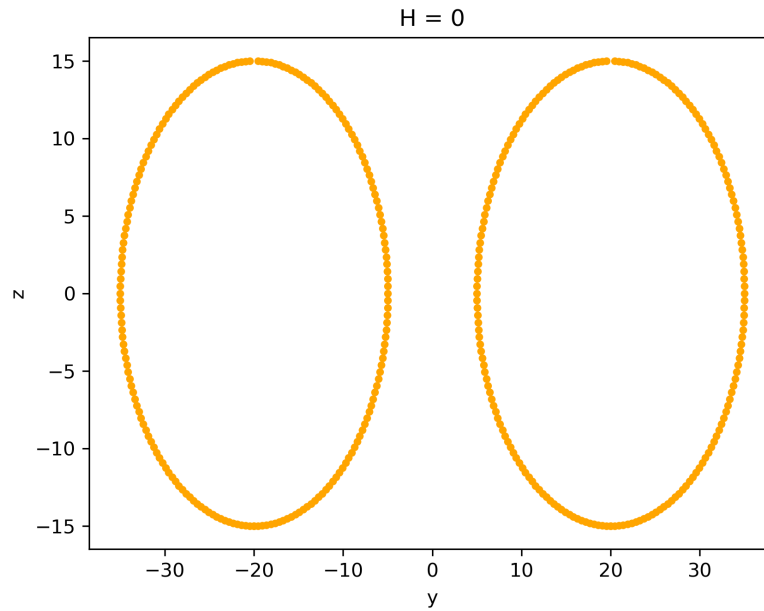


Рис. 2: Сечение тора плоскость  $x = 5$

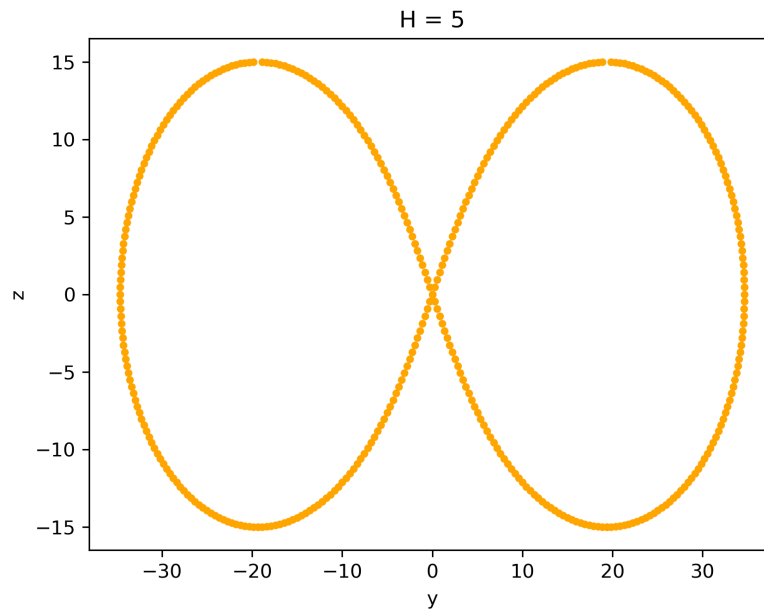


Рис. 3: Сечение тора плоскость  $x = 10$

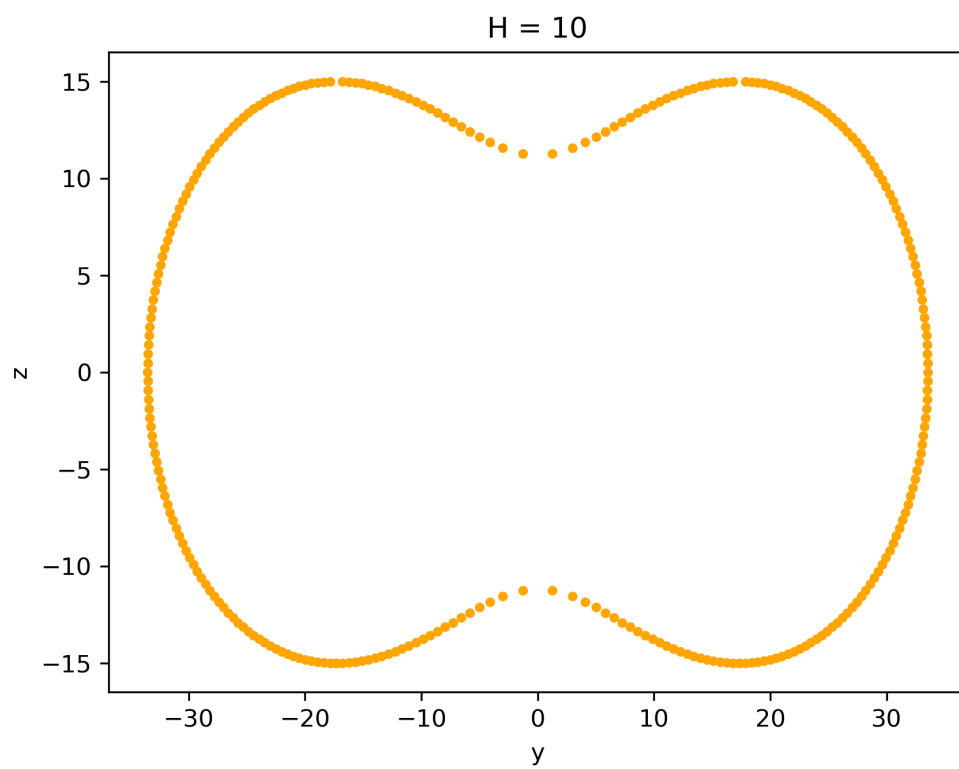


Рис. 4: Сечение тора плоскость  $x = 15$

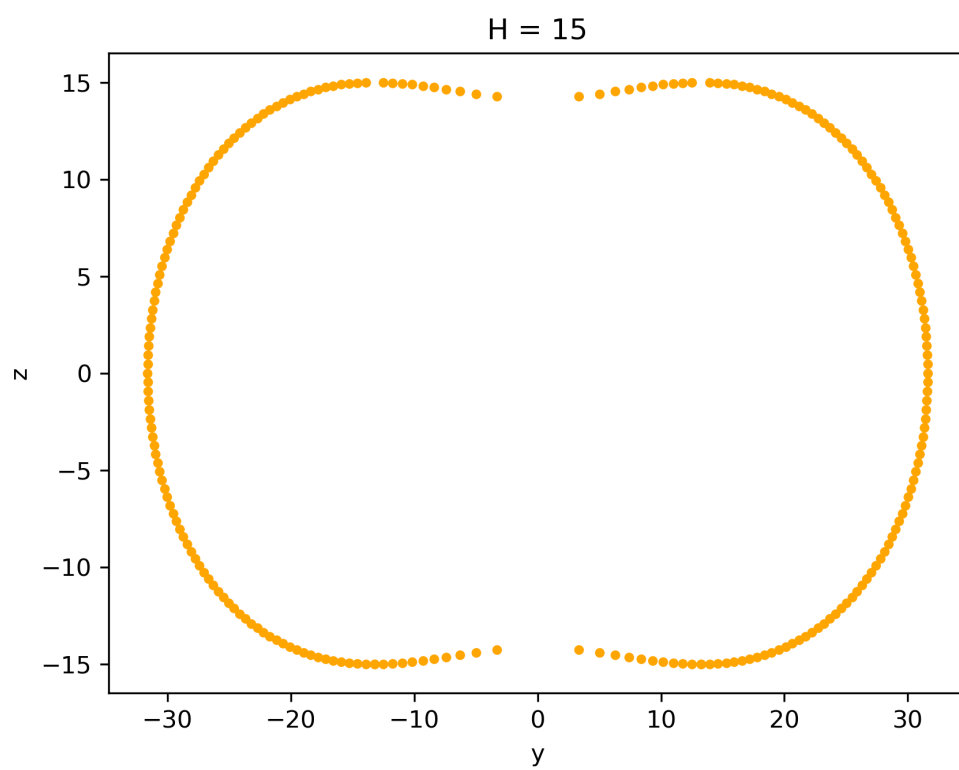


Рис. 5: Сечение тора плоскость  $x = 20$

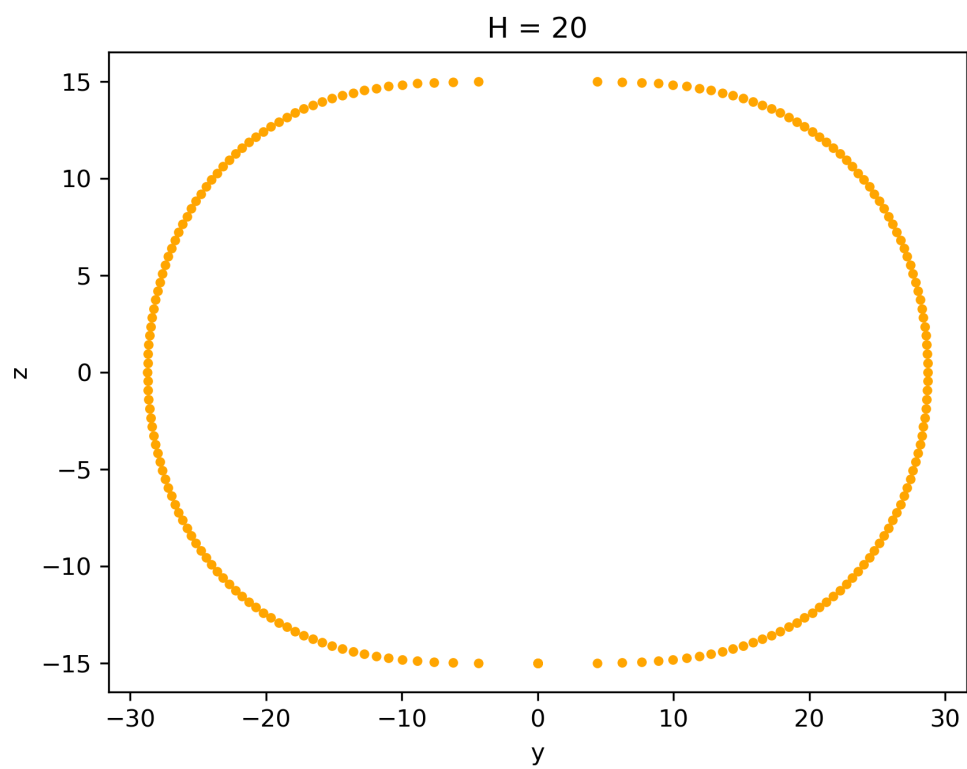


Рис. 6: Сечение тора плоскость  $x = 25$

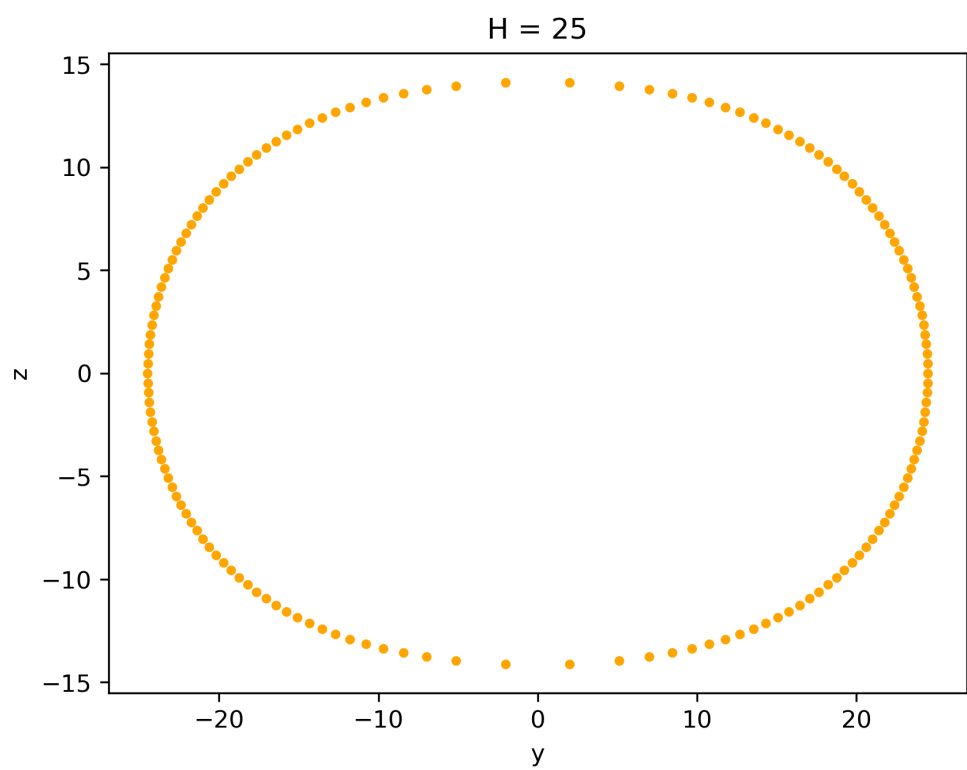


Рис. 7: Сечение тора плоскость  $x = 35$

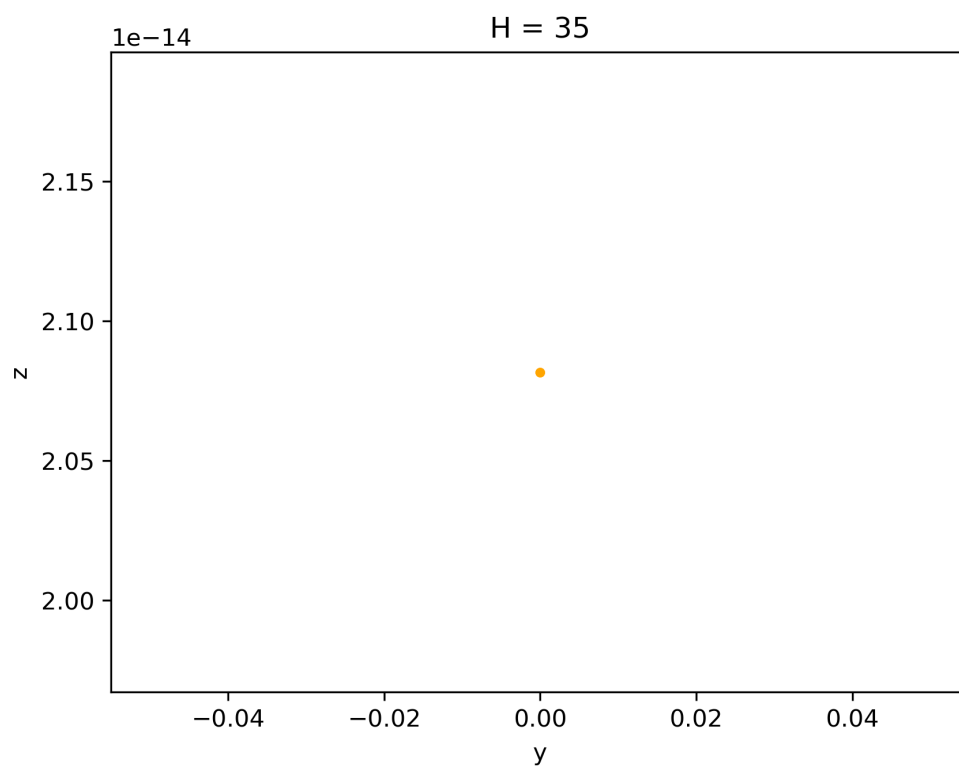
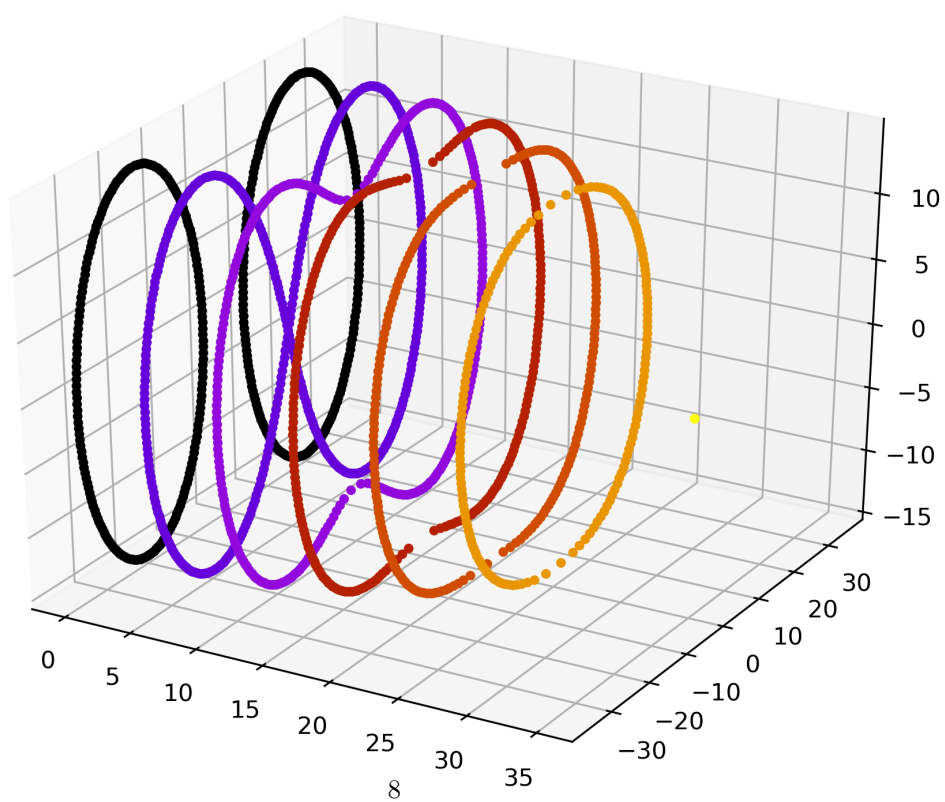


Рис. 8: Все сечения на одном графике





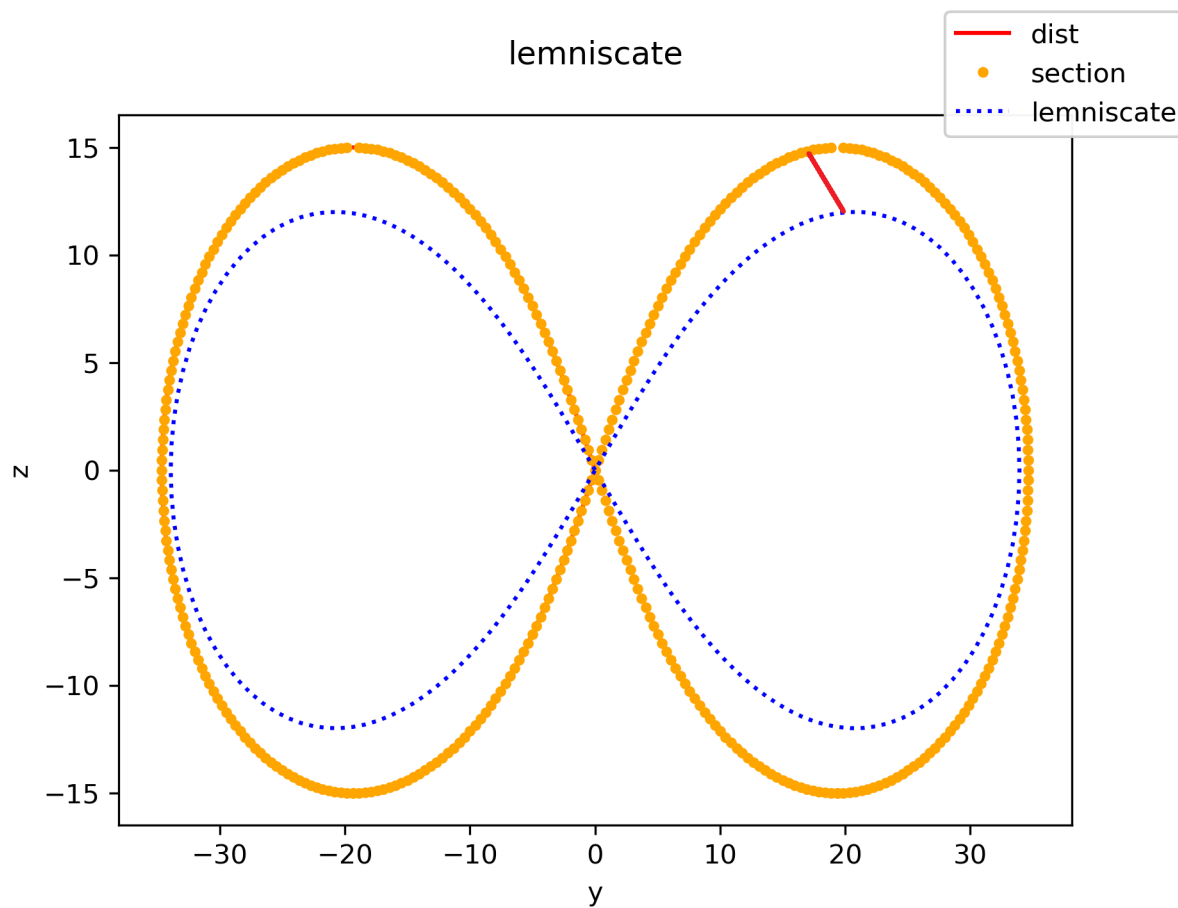
## 5.2 Сравнение с лемниской Бернулли

Рассматривается тор со следующими характеристиками:  $R = 20$  – радиус вращения,  $r = 15$  – радиус образующей  $c = 34$ .

Сечение плоскостью  $x = R - r = 5$  не сильно похоже на лемниску Бернулли.

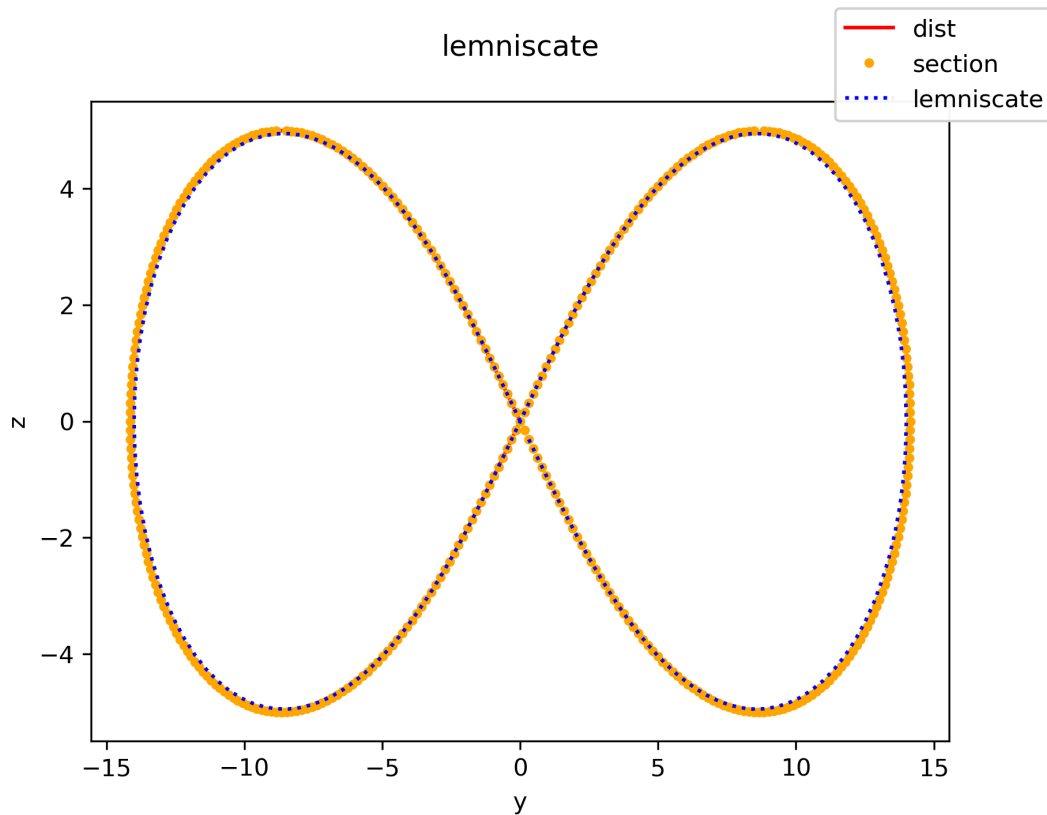
Для их сравнения воспользуемся расстоянием Фреше:  $fr = 4.2426$ .

Рис. 9: Лемниката для сечения тора



Подберём параметры тора так, чтобы лемниката и сечение практически совпадали:  $R = 10$ ,  $r = 5$ , сечение плоскостью  $x = 5$ . Параметр лемнискаты  $c = 14$ . Расстояние Фреше  $fr = 0.1938$ .

Рис. 10: Лемниската практически совпадающая с сечением тора



## 6 Обсуждение

По результатам видно, что в общем случае сечение плоскостью  $x=R-r$  действительно похоже на лемнискату Бернулли, но не совпадает с ним.

Можно подобрать параметры тора и лемнискаты так, чтобы сечение практически совпадало с лемниской (расстояние Фреше мало, но не равняется нулю из-за дискретизации фигур)

Для совпадения радиус образующей тора ( $r$ ) должен быть в 2 раза меньше его радиуса вращения ( $R$ ).

Параметр лемнискаты – параметр масштаба (не влияет на её форму), следовательно, от него не зависит необходимое соотношение между  $r$  и  $R$ .

## 7 Список литературы

- [1] Модуль numpy - <https://physics.susu.ru/vorontsov/language/numpy.html>
- [2] Модуль matplotlib - <https://matplotlib.org/users/index.html>
- [3] Модуль mpl\_toolkit - [https://matplotlib.org/mpl\\_toolkits/mplot3d/api.html](https://matplotlib.org/mpl_toolkits/mplot3d/api.html)
- [4] Пособие к Лабораторным работам <https://cloud.mail.ru/public/4ra6/5wwqBzMBC/LabPractices.pdf>

## 8 Приложения

Код отчёта: [https://github.com/MisterProper9000/computing-complex/tree/master/Lab\\_2\\_body\\_of\\_rotation/lab2.tex](https://github.com/MisterProper9000/computing-complex/tree/master/Lab_2_body_of_rotation/lab2.tex)

Код лабораторной: [https://github.com/MisterProper9000/computing-complex/blob/master/Lab\\_2\\_body\\_of\\_rotation/lab\\_2\\_code/Lab\\_2.py](https://github.com/MisterProper9000/computing-complex/blob/master/Lab_2_body_of_rotation/lab_2_code/Lab_2.py)

```
1 import pylab
2 import frechet
3 import matplotlib.pyplot as plt
4 # noinspection PyInterpreter
5 from mpl_toolkits.mplot3d import Axes3D
6 import numpy as np
7
8
9 class TorPoint:
10     def __init__(self, z_, r1_, r2_):
11         self.z = z_
12         self.r1 = r1_
13         self.r2 = r2_
14
15
16 class Point:
17     def __init__(self, x_, y_, z_):
18         self.x = x_
19         self.y = y_
20         self.z = z_
21
22
23 class Tor:
24     def __init__(self, z_, r_rotate, r_circle):
25         self.z = z_
26         self.R = r_rotate
27         self.r = r_circle
28         self.points = []
29
30     def generate_points(self, num_points):
31         self.points = []
32         n = num_points
33         step = np.pi / n
34         fi = -np.pi / 2.0
35
36         for i in range(0, n):
37             z = self.r * np.sin(fi)
38             dr = self.r * np.cos(fi)
39             point = TorPoint(z, self.R - dr, self.R + dr)
40             self.points.append(point)
41             fi += step
42
43     def get_tor_points(self):
44         return self.points
45
46
47 def points_on_plane(tor_point, x_plane):
48     z = tor_point.z
49     r1 = tor_point.r1
50     r2 = tor_point.r2
51
52     point1 = Point(x_plane, np.sqrt(r1**2 - x_plane**2), z)
53     point2 = Point(x_plane, np.sqrt(r2**2 - x_plane**2), z)
54
55     point3 = Point(x_plane, -np.sqrt(r1**2 - x_plane**2), z)
56     point4 = Point(x_plane, -np.sqrt(r2**2 - x_plane**2), z)
57     return point1, point2, point3, point4
58
```

```

59
60 def get_y_array(points):
61     data = []
62     for point in points:
63         data.append(point.y)
64     return data
65
66
67 def get_z_array(points):
68     data = []
69     for point in points:
70         data.append(point.z)
71     return data
72
73
74 def get_intersection_tor_plane(tor, x_plane):
75     tor_points = tor.get_tor_points()
76
77     result = []
78     left1 = []
79     left2 = []
80     left3 = []
81     left4 = []
82
83     left_jump_2_3_flag = False
84     left_jump_1_4_flag = False
85     right_jump_2_3_flag = False
86     right_jump_1_4_flag = False
87
88     right1 = []
89     right2 = []
90     right3 = []
91     right4 = []
92     for point in tor_points:
93         tmp = points_on_plane(point, x_plane)
94
95         if not np.isnan(tmp[0].y):
96             if not right_jump_1_4_flag:
97                 right1.append(tmp[0])
98             else:
99                 right4.append(tmp[0])
100         else:
101             right_jump_1_4_flag = True
102
103         if not np.isnan(tmp[1].y):
104             if not right_jump_2_3_flag:
105                 right2.append(tmp[1])
106             else:
107                 right3.append(tmp[1])
108         else:
109             right_jump_2_3_flag = True
110
111         if not np.isnan(tmp[2].y):
112             if not left_jump_1_4_flag:
113                 left1.append(tmp[2])
114             else:
115                 left4.append(tmp[2])
116         else:
117             left_jump_1_4_flag = True
118
119         if not np.isnan(tmp[3].y):
120             if not left_jump_2_3_flag:
121                 left2.append(tmp[3])
122             else:
123                 left3.append(tmp[3])
124         else:

```

```

125         left_jump_2_3_flag = True
126
127         right1.reverse()
128         right4.reverse()
129         left1.reverse()
130         left4.reverse()
131
132         left = left2 + left3 + left4 + left1
133         right = right2 + right3 + right4 + right1
134         right.reverse()
135
136         return left, right
137
138
139 def print_points(points):
140     print()
141     for point in points:
142         print(point.x, point.y, point.z)
143
144
145 def plot_3D(plane_cut, name, filename):
146
147     data = []
148
149     fig = pylab.figure()
150     axes = Axes3D(fig)
151
152     i = 0
153     number = len(plane_cut)
154     cmap = plt.get_cmap('gnuplot')
155     colors = [cmap(i) for i in np.linspace(0, 1, number)]
156     for points_arr in plane_cut:
157         for points in points_arr:
158             x = []
159             y = []
160             z = []
161             x1 = []
162             y1 = []
163             z1 = []
164             for elem in points:
165                 x.append([elem.x])
166                 y.append([elem.y])
167                 z.append([elem.z])
168                 x1.append(elem.x)
169                 y1.append(elem.y)
170                 z1.append(elem.z)
171
172             x = np.matrix(x)
173             y = np.matrix(y)
174             z = np.matrix(z)
175             axes.plot(x1, y1, z1, ".", color=colors[i])
176             #axes.plot_wireframe(x, y, z, color=colors[i], linewidth=5,
177             linestyle='-')
178
179             i += 1
180
181     fig.savefig(filename, dpi=300, format='png', bbox_inches='tight')
182     fig.show()
183     plt.close(fig)
184
185 def draw_cut(points_arr, name, filename):
186     plt.axis('scaled')
187     fig, ax = plt.subplots(nrows=1, ncols=1, sharey=True)
188     ax.set_xlabel("y")
189     ax.set_ylabel("z")

```

```

190     ax.set_title(name)
191
192     for points in points_arr:
193         x = []
194         y = []
195         z = []
196         for elem in points:
197             x.append(elem.x)
198             y.append(elem.y)
199             z.append(elem.z)
200         ax.plot(y, z, ".", color="orange")
201     box = ax.get_position()
202     ax.set_position([box.x0, box.y0, box.width, box.height])
203     #fig.legend()
204     fig.savefig(filename, dpi=300, format='png', bbox_inches='tight')
205
206     fig.show()
207     plt.close(fig)
208
209
210 def plane_research(tor, planes):
211     data = []
212     for plane in planes:
213         left, right = get_intersection_tor_plane(tor, plane)
214         draw_cut([left, right], "H = %i" % plane, "Tor_section(H = %i).png" %
215                 plane)
216         data.append([left, right])
217
218     plot_3D(data, "H = %i" % len(planes), "Tor_cuts(H = %i).png" % len(planes))
219
220 def lemniscate(c):
221     result = []
222     n = 360
223     fi = 0
224     step = 2 * np.pi / n
225     left = []
226     right = []
227     for i in range(0, n//4):
228         t = np.tan(fi)
229         x = c * (t + t ** 3) / (1 + t ** 4)
230         y = c * (t - t ** 3) / (1 + t ** 4)
231         right.append([x, y])
232         fi += step
233
234     for i in range(n//4, n // 2):
235         t = np.tan(fi)
236         x = c * (t + t ** 3) / (1 + t ** 4)
237         y = c * (t - t ** 3) / (1 + t ** 4)
238         left.append([x, y])
239         fi += step
240
241     return left, right
242
243
244 def draw_line_on_plot(line, ax, color):
245     data = [line]
246     for i in range(0, len(data)):
247         x = []
248         y = []
249         for elem in data[i]:
250             x.append(elem[0])
251             y.append(elem[1])
252         ax.plot(x, y, ":", label="lemniscate", color=color)
253

```

```

254
255 def process_lemniscate(tor, plane, c):
256     name = "lemniscate"
257     filename = "lemniscate.png"
258
259
260     left, right = get_intersection_tor_plane(tor, plane)
261     points_arr = [left, right]
262     l, r = lemniscate(c)
263
264
265     plt.axis('scaled')
266     fig, ax = plt.subplots(nrows=1, ncols=1, sharey=True)
267     ax.set_xlabel("y")
268     ax.set_ylabel("z")
269     ax.set_title(name + "\n")
270
271     data = []
272     data_arr = []
273     draw_flag = True
274     for points in points_arr:
275         x = []
276         y = []
277         z = []
278         tmp = []
279         for elem in points:
280             x.append(elem.x)
281             y.append(elem.y)
282             z.append(elem.z)
283             data.append([elem.y, elem.z])
284             tmp.append([elem.y, elem.z])
285         data_arr.append(tmp)
286         if(draw_flag):
287             ax.plot(y, z, "-", color="red", label="dist")
288             ax.plot(y, z, ".", color="orange", label="section")
289             draw_flag = False
290         else:
291             ax.plot(y, z, ".", color="orange")
292
293     draw_line_on_plot(l + r, ax, "b")
294
295     temp_lemn = [r, l]
296     temp_data = [data_arr[0], data_arr[1]]
297     dist1, ind1 = frechet.d_Frechet(temp_lemn[0], temp_data[1])
298     dist2, ind2 = frechet.d_Frechet(temp_lemn[0], temp_data[1])
299
300     leg = fig.legend()
301     fig.legend()
302     fig.savefig(filename, dpi=300, format='png', bbox_inches='tight')
303
304     fig.show()
305     plt.close(fig)
306
307
308 def main():
309     R = 10
310     r1 = 5
311     numpoints = 360
312
313     tor = Tor(0, R, r1)
314     tor.generate_points(100)
315     H = [0, 5, 10, 15, 20, 25, 35]
316     plane_research(tor, H)
317     process_lemniscate(tor, R - r1, R + r1 - r1/5)
318
319     print("plane = ", R - r1, "c = ", 28)

```

```
320
321
322 if __name__ == "__main__":
323     main()
```