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

1 Список иллюстраций

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 постоить 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:

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

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]

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

где $t = \mathrm{tg}(\varphi), c$ – 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 $x = 0$

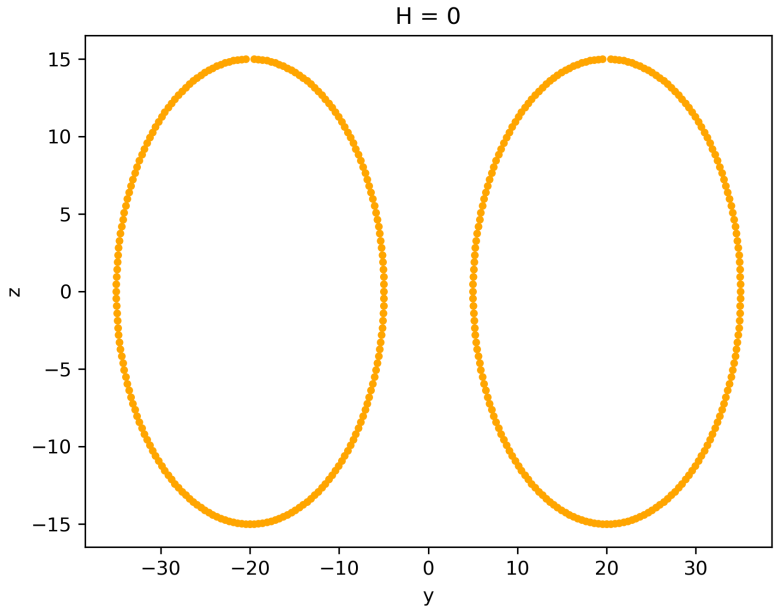


Рис. 2: Сечение тора плоскость $x = 5$

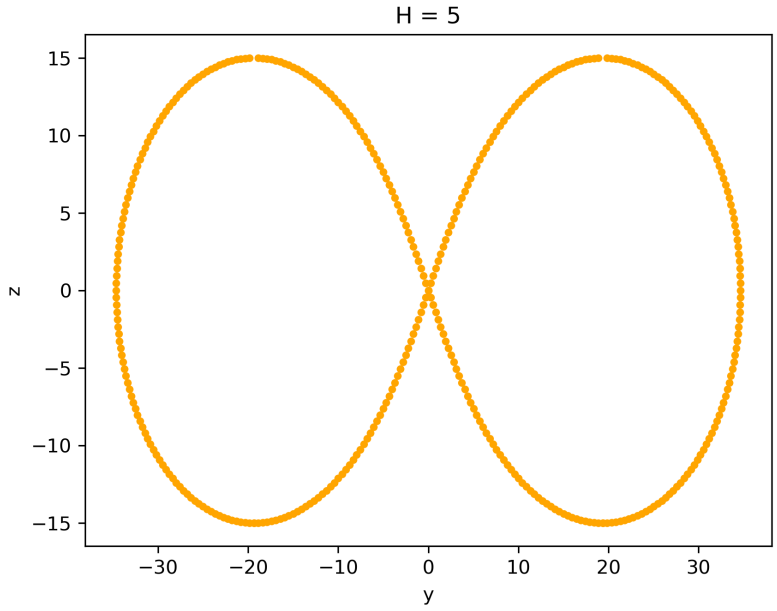


Рис. 3: Сечение тора плоскость $x = 10$

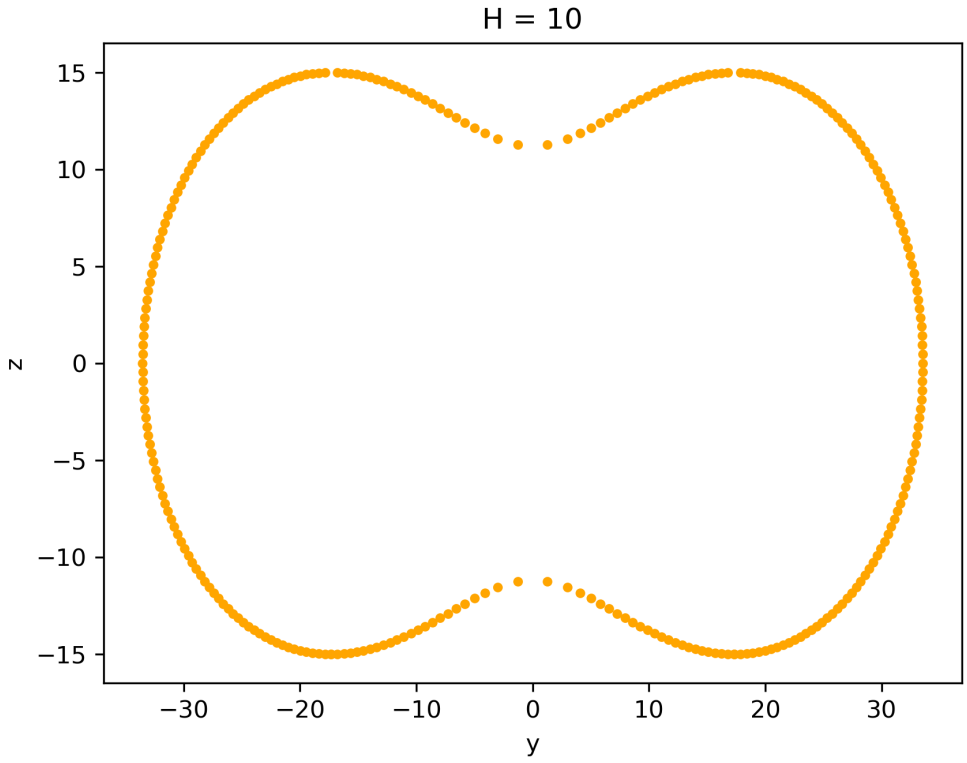


Рис. 4: Сечение тора плоскость $x = 15$

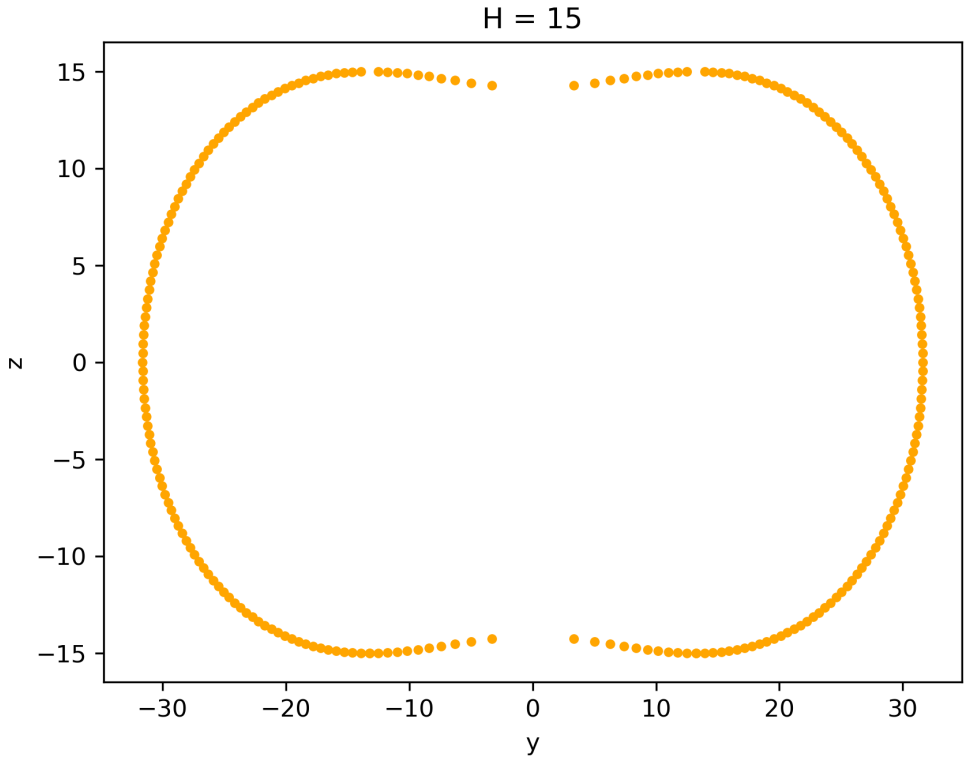


Рис. 5: Сечение тора плоскость $x = 20$

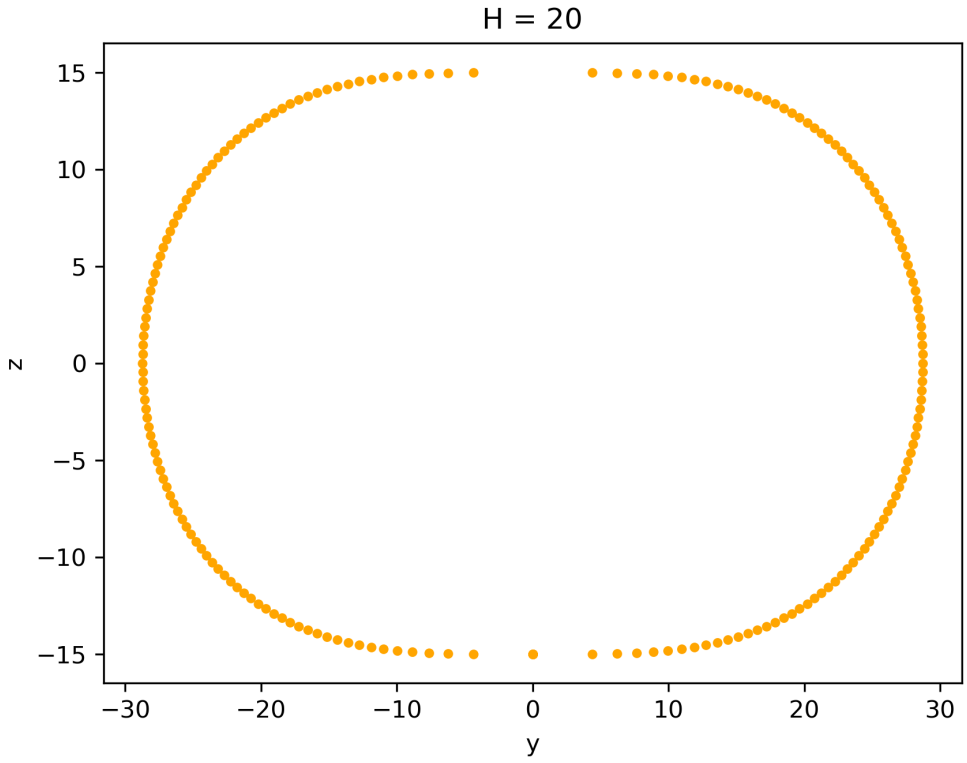


Рис. 6: Сечение тора плоскость $x = 25$

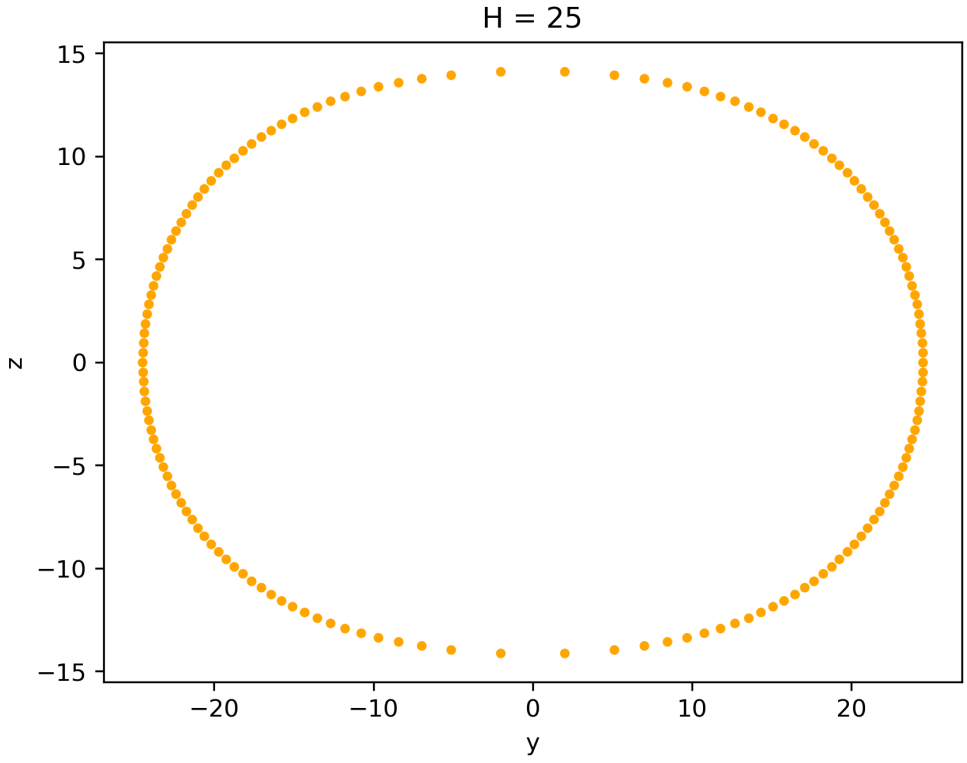


Рис. 7: Сечение тора плоскость $x = 30$

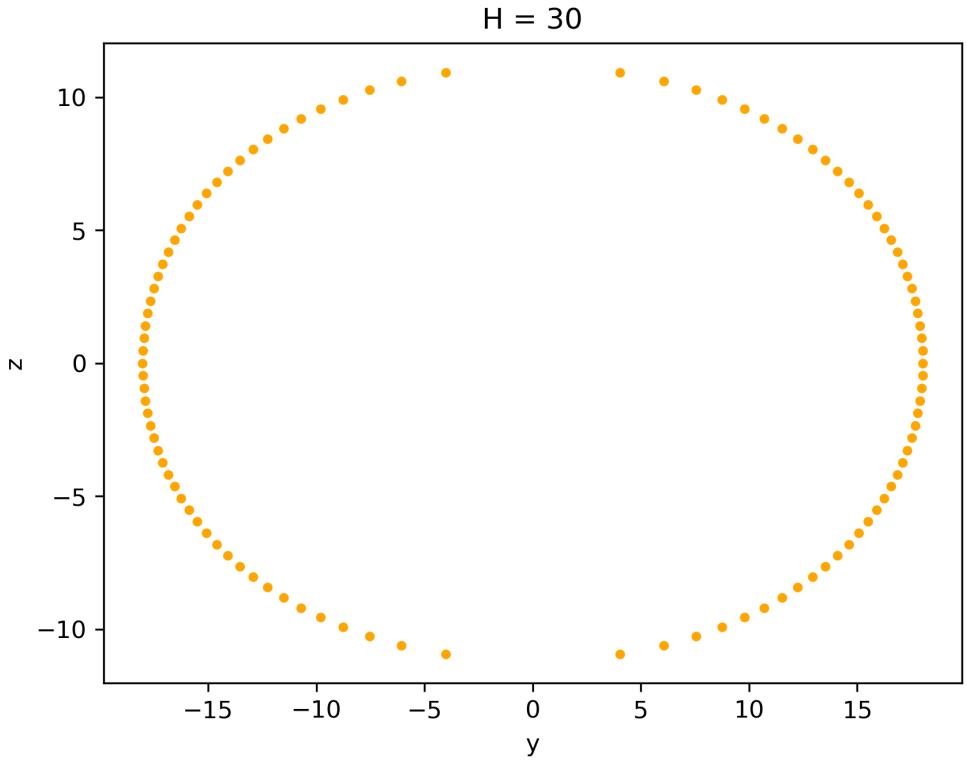


Рис. 8: Сечение тора плоскость $x = 35$

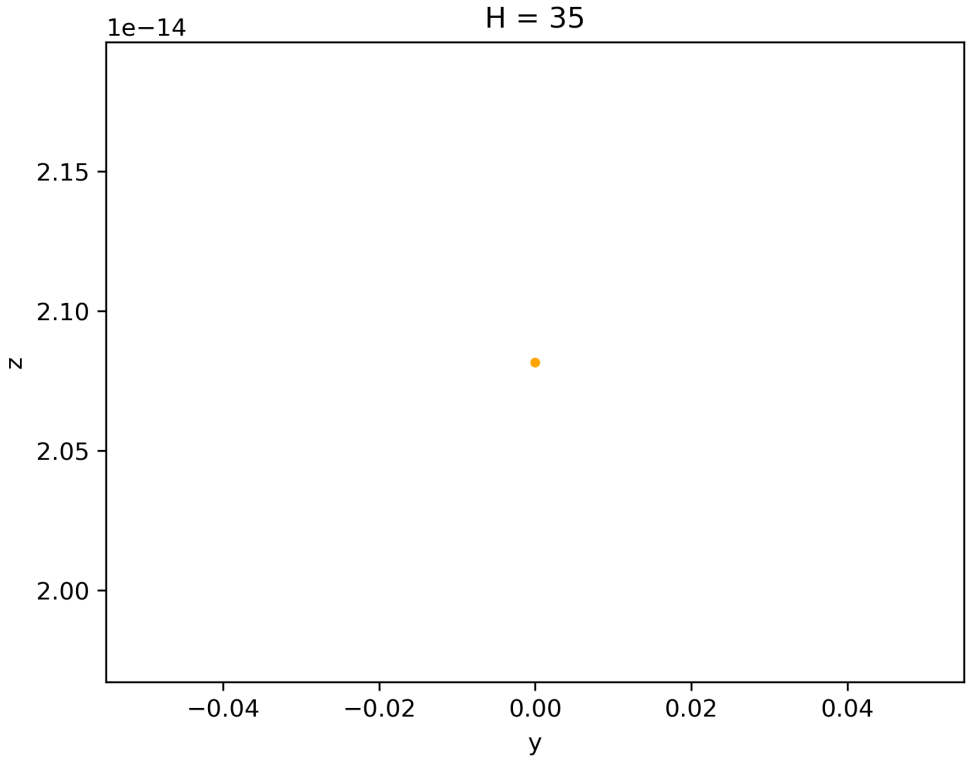
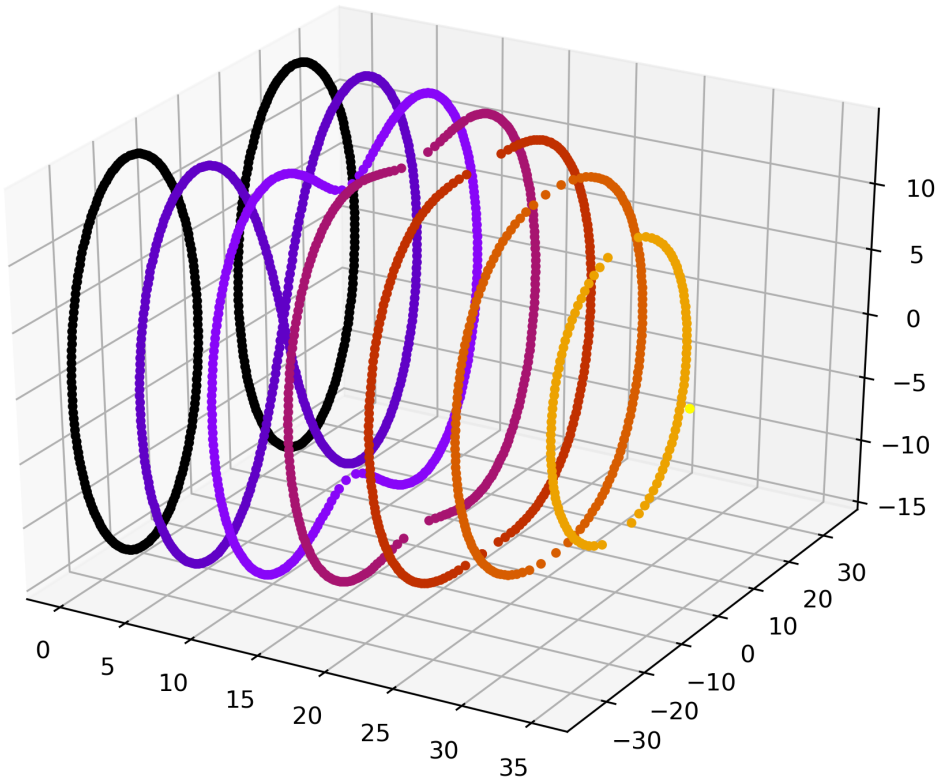


Рис. 9: Все сечения на одном графике



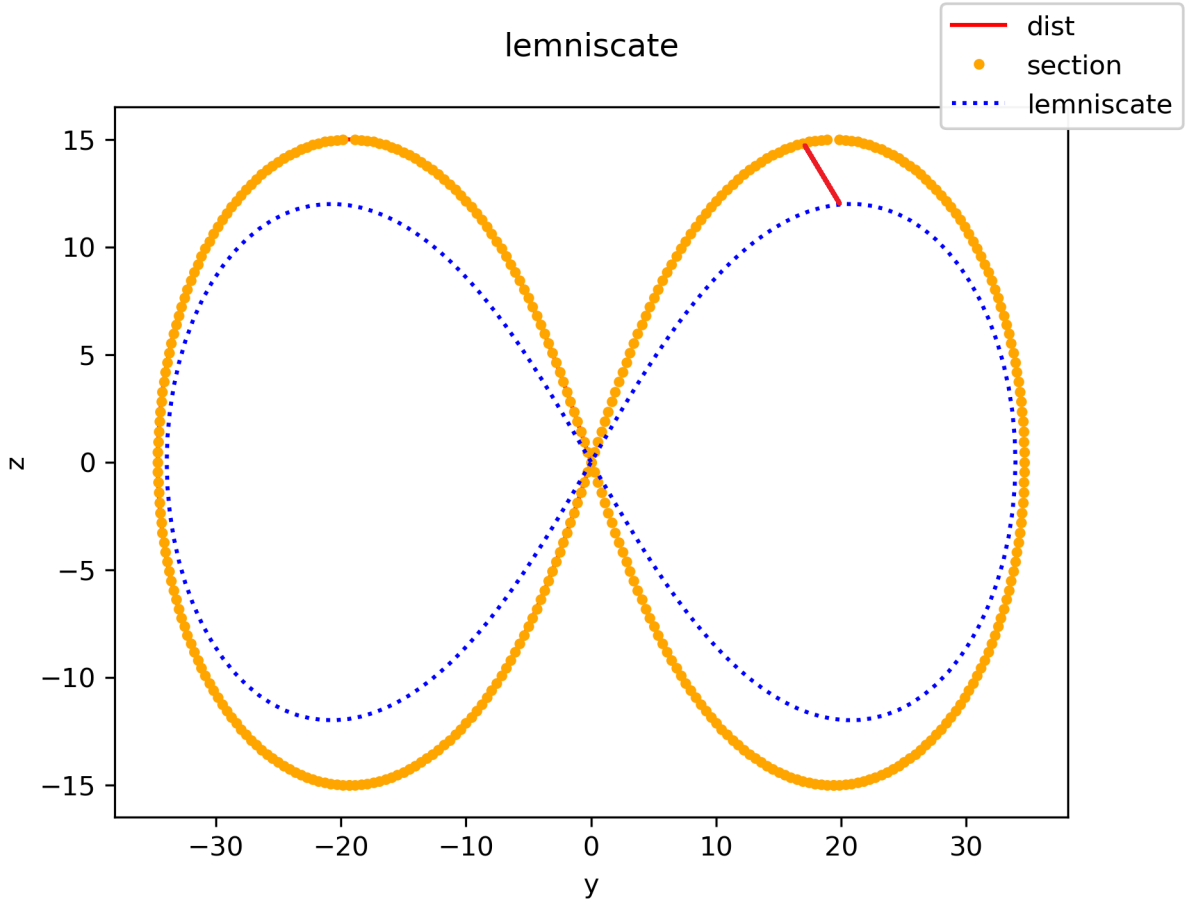
5.2 Сравнение с лемнискатой Бернулли

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 лемнискату Бернулли.

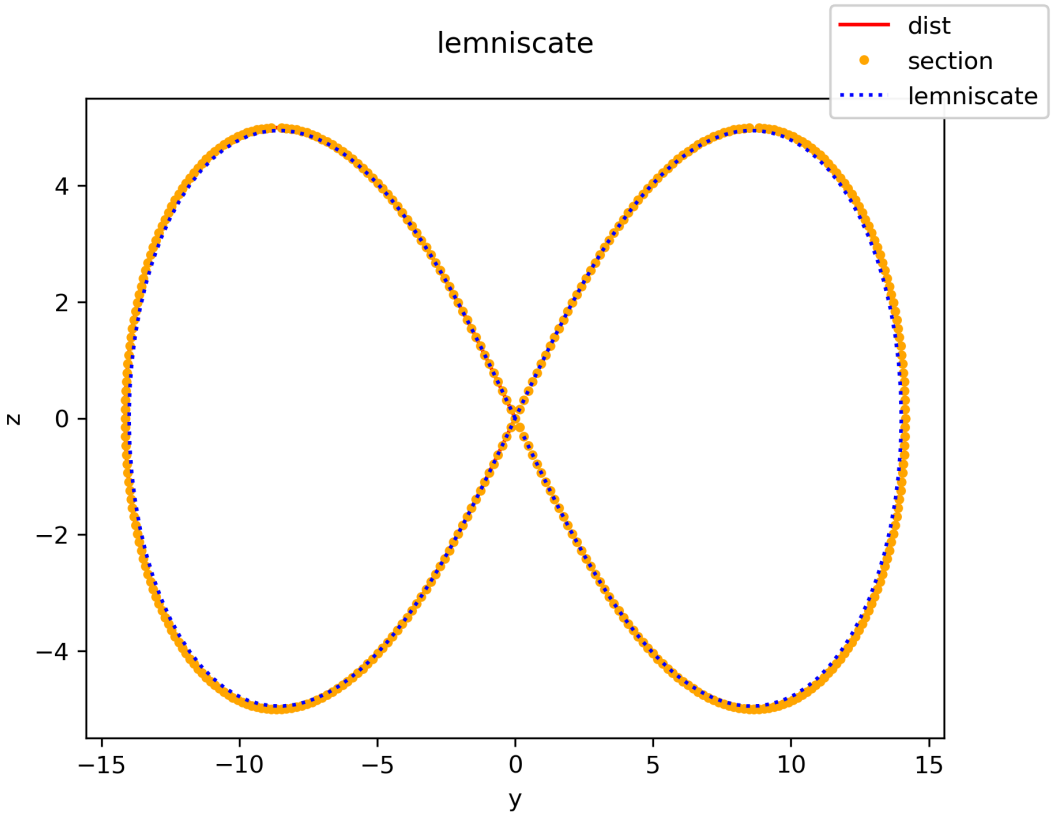
Для их сравнения воспользуемся расстоянием Фреше: $fr = 4.2426$.

Рис. 10: Лемниската для сечения тора



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

Рис. 11: Лемниската практически совпадающая с сечением тора



6 Обсуждение

По результатам видно, что общем случае сечение плоскость 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 лемнискатой (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

[4] Пособие к Лабораторным работам https://cloud.mail.ru/public/4ra6/5wwqBzMBC/LabPractics.pdf

8 Приложения

Код отчёта: 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

```
import pylab
import frechet
import matplotlib.pyplot as plt
# noinspection PyInterpreter
from mpl_toolkits.mplot3d import Axes3D
import numpy as np


class TorPoint:
    def __init__(self, z_, r1_, r2_):
        self.z = z_
        self.r1 = r1_
        self.r2 = r2_


class Point:
    def __init__(self, x_, y_, z_):
        self.x = x_
        self.y = y_
        self.z = z_


class Tor:
    def __init__(self, z_, r_rotate, r_circle):
        self.z = z_
        self.R = r_rotate
        self.r = r_circle
        self.points = []

    def generate_points(self, num_points):
        self.points = []
        n = num_points
        step = np.pi / n
        fi = -np.pi / 2.0

        for i in range(0, n):
            z = self.r * np.sin(fi)
            dr = self.r * np.cos(fi)
            point = TorPoint(z, self.R - dr, self.R + dr)
            self.points.append(point)
            fi += step

    def get_tor_points(self):
        return self.points


def points_on_plane(tor_point, x_plane):
    z = tor_point.z
    r1 = tor_point.r1
    r2 = tor_point.r2

    point1 = Point(x_plane, np.sqrt(r1**2 - x_plane**2), z)
    point2 = Point(x_plane, np.sqrt(r2**2 - x_plane**2), z)

    point3 = Point(x_plane, -np.sqrt(r1 ** 2 - x_plane ** 2), z)
    point4 = Point(x_plane, -np.sqrt(r2 ** 2 - x_plane ** 2), z)
    return point1, point2, point3, point4

```

```
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:
```

```
            left_jump_2_3_flag = True

    right1.reverse()
    right4.reverse()
    left1.reverse()
    left4.reverse()

    left = left2 + left3 + left4 + left1
    right = right2 + right3 + right4 + right1
    right.reverse()

    return left, right


def print_points(points):
    print()
    for point in points:
        print(point.x, point.y, point.z)


def plot_3D(plane_cut, name, filename):

    data = []

    fig = pylab.figure()
    axes = Axes3D(fig)

    i = 0
    number = len(plane_cut)
    cmap =plt.get_cmap('gnuplot')
    colors = [cmap(i) for i in np.linspace(0, 1, number)]
    for points_arr in plane_cut:
        for points in points_arr:
            x = []
            y = []
            z = []
            x1 = []
            y1 = []
            z1 = []
            for elem in points:
                x.append([elem.x])
                y.append([elem.y])
                z.append([elem.z])
                x1.append(elem.x)
                y1.append(elem.y)
                z1.append(elem.z)

            x = np.matrix(x)
            y = np.matrix(y)
            z = np.matrix(z)
            axes.plot(x1, y1, z1, ".", color=colors[i])
            #axes.plot_wireframe(x, y, z, color=colors[i], linewidth=5,
    linestyle='-')

        i += 1

    fig.savefig(filename, dpi=300, format='png', bbox_inches='tight')
    fig.show()
    plt.close(fig)


def draw_cut(points_arr, name, filename):
    plt.axis('scaled')
    fig, ax = plt.subplots(nrows=1, ncols=1, sharey=True)
    ax.set_xlabel("y")
    ax.set_ylabel("z")
```

```
    ax.set_title(name)

    for points in points_arr:
        x = []
        y = []
        z = []
        for elem in points:
            x.append(elem.x)
            y.append(elem.y)
            z.append(elem.z)
        ax.plot(y, z, ".", color="orange")
    box = ax.get_position()
    ax.set_position([box.x0, box.y0, box.width, box.height])
    #fig.legend()
    fig.savefig(filename, dpi=300, format='png', bbox_inches='tight')

    fig.show()
    plt.close(fig)


def plane_research(tor, planes):
    data = []
    for plane in planes:
        left, right = get_intersection_tor_plane(tor, plane)
        draw_cut([left, right], "H = %i" % plane, "Tor_section(H = %i).png" %
     plane)
        data.append([left, right])

    plot_3D(data, "H = %i" % len(planes), "Tor_cuts(H = %i).png" % len(planes
    ))


def lemniscate(c):
    result = []
    n = 360
    fi = 0
    step = 2 * np.pi / n
    left = []
    right = []
    for i in range(0, n//4):
        t = np.tan(fi)
        x = c * (t + t ** 3) / (1 + t ** 4)
        y = c * (t - t ** 3) / (1 + t ** 4)
        right.append([x, y])
        fi += step

    for i in range(n//4, n // 2):
        t = np.tan(fi)
        x = c * (t + t ** 3) / (1 + t ** 4)
        y = c * (t - t ** 3) / (1 + t ** 4)
        left.append([x, y])
        fi += step

    return left, right


def draw_line_on_plot(line, ax, color):
    data = [line]
    for i in range(0, len(data)):
        x = []
        y = []
        for elem in data[i]:
            x.append(elem[0])
            y.append(elem[1])
        ax.plot(x, y, ":", label="lemniscate", color=color)

```

```
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 = 20
310     r1 = 15
311     numpoints = 360
312
313     tor = Tor(0, R, r1)
314     tor.generate_points(100)
315     H = [0, 5, 10, 15, 20, 25, 30, 35]
316     plane_research(tor, H)
317     process_lemniscate(tor, R - r1, R + r1 - r1/15)
318
319     print("plane = ", R - r1, "c = ", 28)
```

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