|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 10**

|  |  |
| --- | --- |
| **Тема: Алгоритм плавающего горизонта**  **Студент** ВоякинА. Я.  **Группа ИУ7-44Б**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |  |

Москва.

2020 г.

**Цель работы:**

Изучение и программная реализация алгоритма плавающего горизонта построения трёхмерных поверхностей

**Условие задания:**

Должна быть разработана программа, позволяющая осуществлять ввод пределов и шага изменения координат x, z, выбора уравнения поверхности из заранее сформированного списка, построение поверхности. Должен быть обеспечен поворот изображения (поверхности) вокруг каждой из трех координатных осей. Система координат должна быть неподвижной. Выполнить масштабирование для обеспечения размещения исходного изображения целиком в пределах поля вывода. Список уравнений поверхностей задается в отдельном модуле.

**Описание алгоритма:**

Алгоритм плавающего горизонта предназначен для построения поверхностей, заданных неявных уравнением F(x, y, z) = 0.

Идея алгоритма состоит в том, что мы заданную поверхность рассекаем плоскостями, перпендикулярными оси Z. Эти плоскости имеют уравнения Z = const.

Рассматриваемая поверхность рассекается плоскостями, перпендикулярными оси Z. В каждом сечении получается кривая, описываемая уравнением y = f(x, z = const).

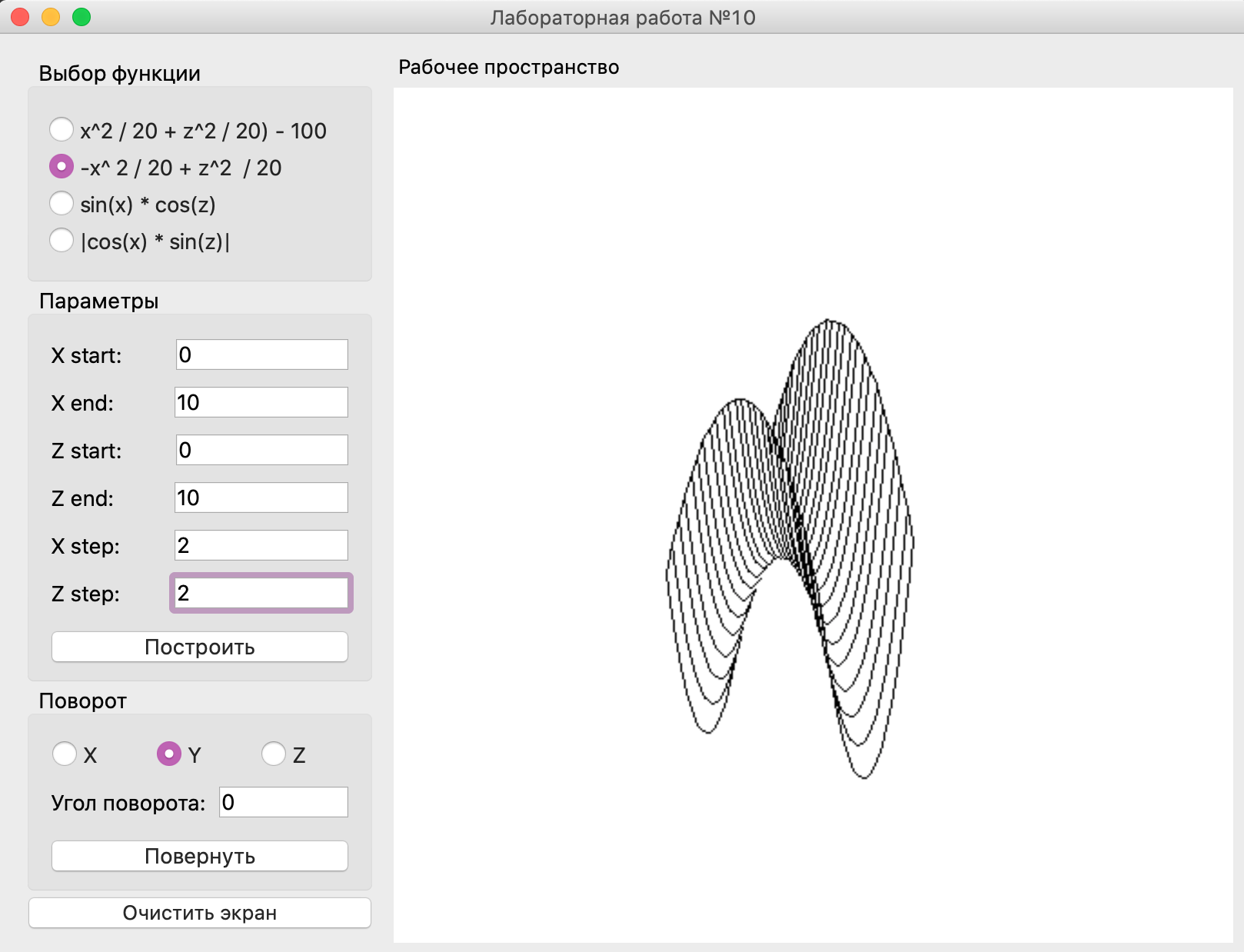
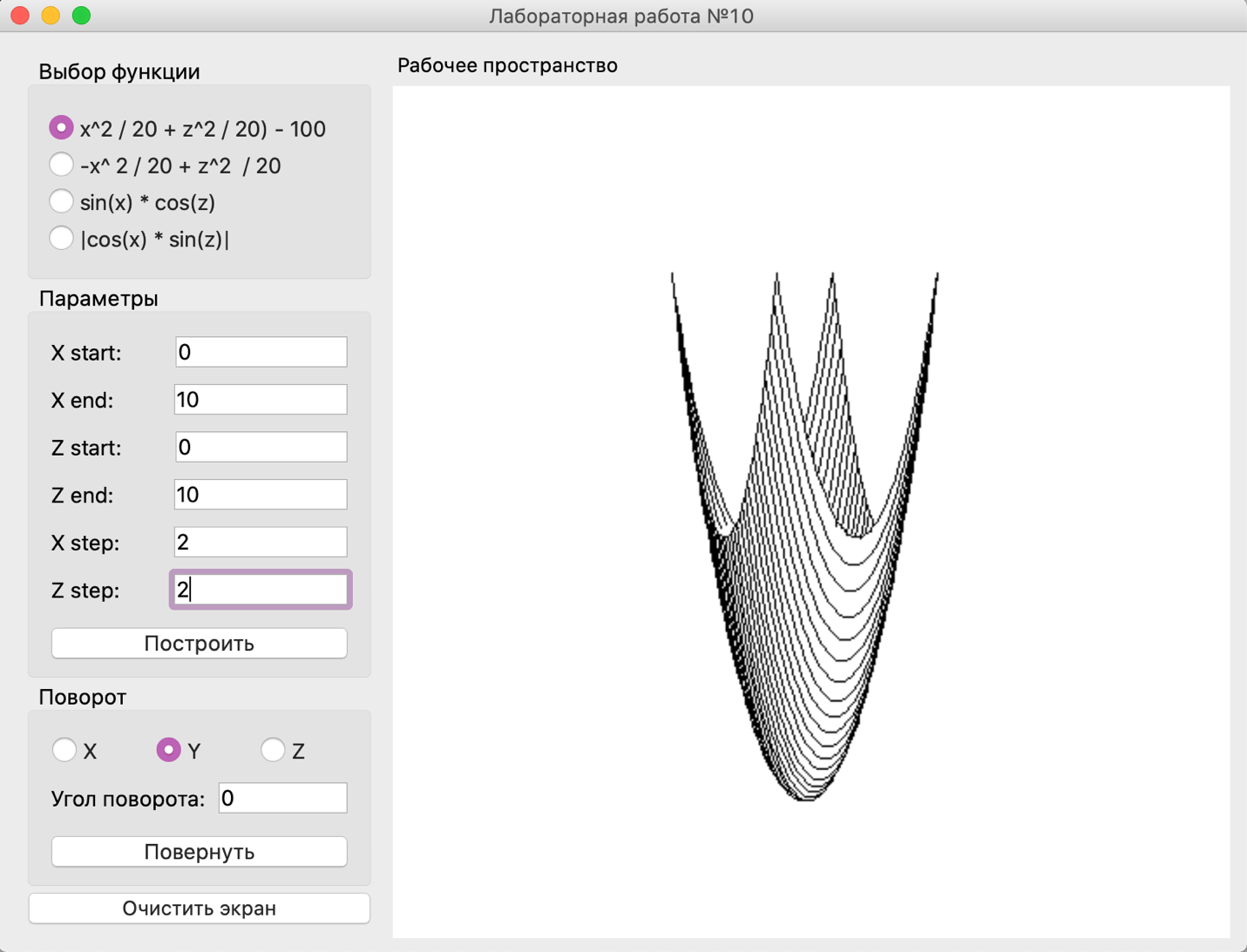
Полученные кривые можно спроецировать на плоскость z = 0. Изобразить видимые части каждой кривой. Изображение надо строить, начиная с кривой, полученной в ближайшем к наблюдателю сечении.

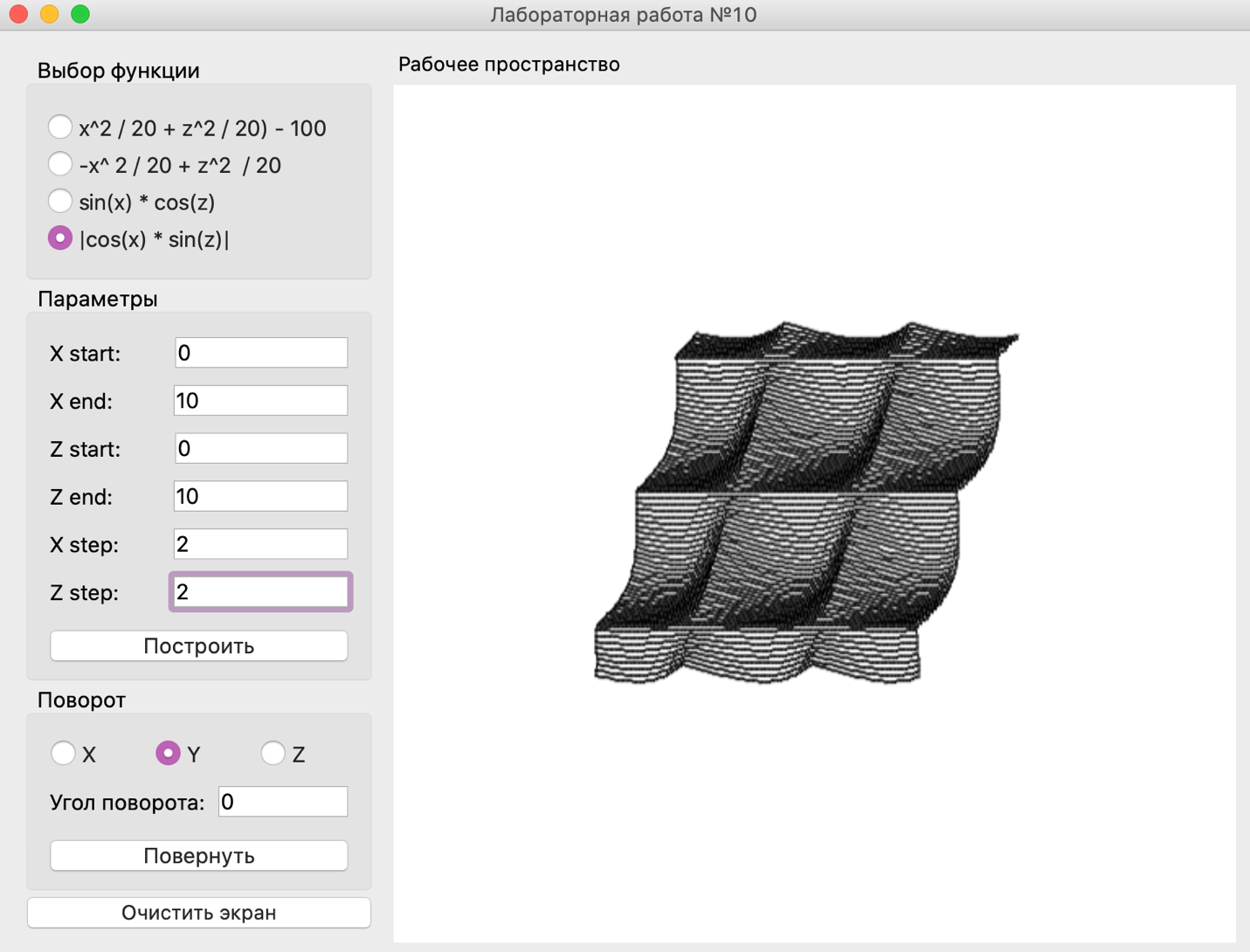
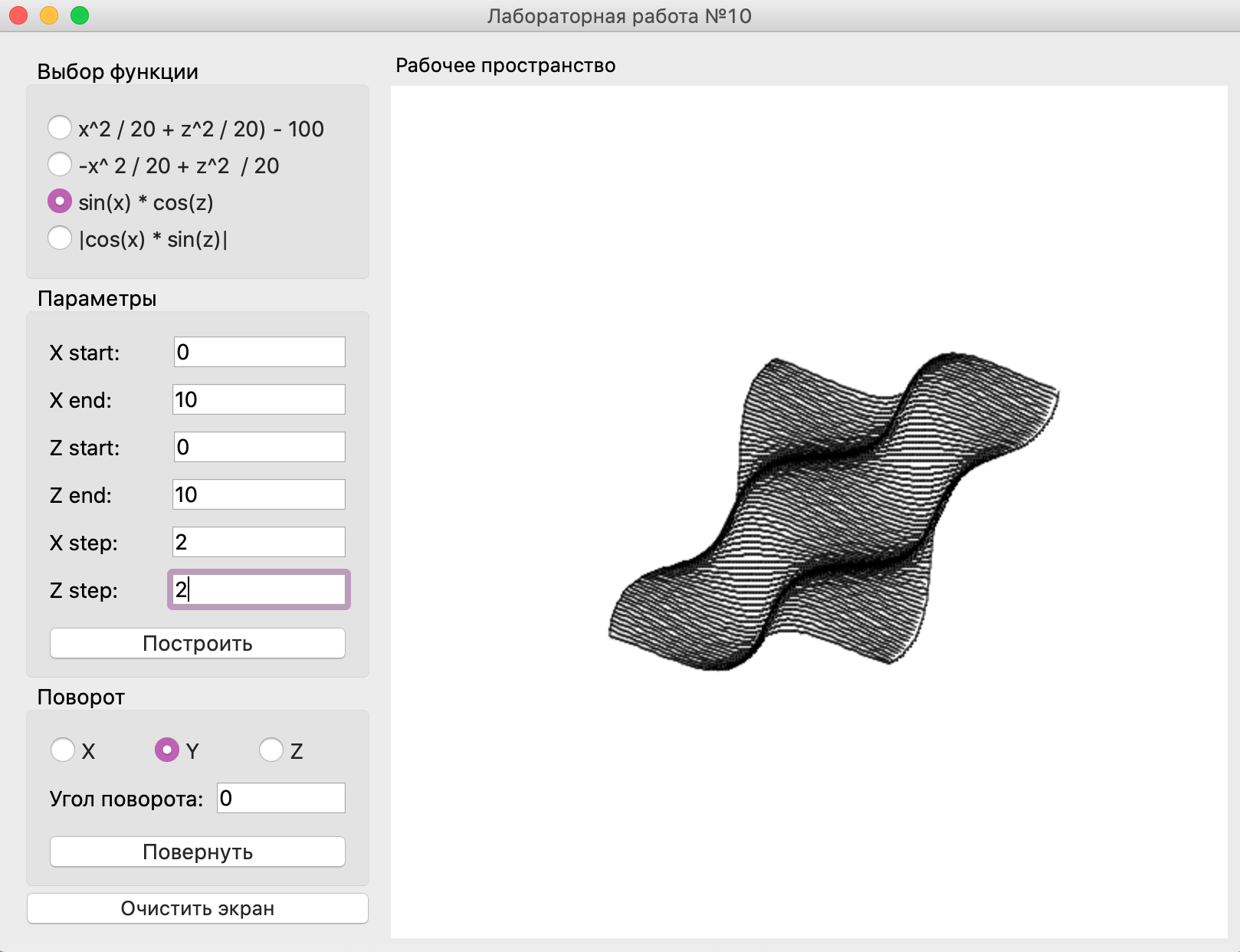
Кривая, полученная в сечении ближайшей к наблюдателю плоскостью, является видимой.

**Код алгоритма.**

def hor(*self*, x1, y1, x2, y2, top, bottom):  
 if x2 - x1 == 0:  
 top[x2] = max(top[x2], y2)  
 bottom[x2] = min(bottom[x2], y2)  
 else:  
 t = (y2 - y1) / (x2 - x1)  
 for x in range(x1, x2 + 1):  
 y = t \* (x - x1) + y1  
 top[x] = max(top[x], y)  
 bottom[x] = min(bottom[x], y)  
  
def obr\_r(*self*, x, y, x\_reb, y\_reb, top, bottom):  
 if x\_reb != -1:  
 *self*.hor(x\_reb, y\_reb, x, y, top, bottom)  
 return x, y  
  
def visability(*self*, x, y, top, bottom):  
 if bottom[x] < y < top[x]:  
 return 0  
 elif y >= top[x]:  
 return 1  
 elif y <= bottom[x]:  
 return -1  
  
def crossing(*self*, x1, y1, x2, y2, hor):  
 if x2 - x1 == 0:  
 xi = x2  
 yi = hor[x2]  
 else:  
 t = (y2 - y1) / (x2 - x1)  
 y\_sign = sign(y1 + t - hor[x1 + 1])  
 c\_sign = y\_sign  
 yi = y1 + t  
 xi = x1 + 1  
 while c\_sign == y\_sign:  
 yi += t  
 xi += 1  
 c\_sign = sign(yi - hor[xi])  
 if fabs(yi - t - hor[xi - 1]) <= fabs(yi - hor[xi]):  
 yi -= t  
 xi -= 1  
 return xi, yi  
  
def AlgHor(*self*):  
 h = 556  
 top = [0]  
 bottom = [h]  
 x\_left = -1  
 y\_left = -1  
 x\_right = -1  
 y\_right = -1  
 for z in range(int(*self*.lineEditZend.text()), int(*self*.lineEditZstart.text()) - 1, -int(*self*.lineEditZstep.text())):  
 x\_prev = int(*self*.lineEditXstart.text())  
 y\_prev = *self*.f(x\_prev, z)  
 x\_prev, y\_prev = *self*.obr\_r(x\_prev, y\_prev, x\_left, y\_left, top, bottom)  
 flag\_vis = *self*.visability(x\_prev, y\_prev, top, bottom)  
 for x in range(int(*self*.lineEditZstart.text()), int(*self*.lineEditXend.text()), int(*self*.lineEditXstep.text())):  
 y = *self*.f(x, z)  
 flag\_t = *self*.visability(x, y, top, bottom)  
 if flag\_t == flag\_vis:  
 if flag\_t == 1 or flag\_t == -1:  
 *self*.bres\_int(x\_prev, y\_prev, x, y, *self*.black)  
 *self*.hor(x\_prev, y\_prev, x, y, top, bottom)  
 else:  
 if flag\_t == 0:  
 if flag\_vis == 1:  
 xi, yi = *self*.crossing(x\_prev, y\_prev, x, y, top)  
 else:  
 xi, yi = *self*.crossing(x\_prev, y\_prev, x, y, bottom)  
 *self*.bres\_int(x\_prev, y\_prev, xi, yi, *self*.black)  
 *self*.hor(x\_prev, y\_prev, xi, yi, top, bottom)  
 else:  
 if flag\_t == 1:  
 if flag\_vis == 0:  
 xi, yi = *self*.crossing(x\_prev, y\_prev, x, y)  
 *self*.bres\_int(x\_prev, y\_prev, xi, yi, *self*.black)  
 *self*.hor(xi, yi, x, y, top, bottom)  
 else:  
 xi, yi = *self*.crossing(x\_prev, y\_prev, x, y, bottom)  
 *self*.bres\_int(x\_prev, y\_prev, xi, yi, *self*.black)  
 *self*.hor(x\_prev, y\_prev, xi, yi, top, bottom)  
 xi, yi = *self*.crossing(x\_prev, y\_prev, x, y, top)  
 *self*.bres\_int(xi, yi, x, y, *self*.black)  
 *self*.hor(xi, yi, x, y, top, bottom)  
 else:  
 if flag\_vis == 0:  
 xi, yi = *self*.crossing(x\_prev, y\_prev, x, y, top)  
 *self*.bres\_int(xi, yi, x, y, *self*.black)  
 *self*.hor(xi, yi, x, y, top, bottom)  
 else:  
 xi, yi = *self*.crossing(x\_prev, y\_prev, x, y, top)  
 *self*.bres\_int(x\_prev, y\_prev, xi, yi, *self*.black)  
 *self*.hor(x\_prev, y\_prev, xi, yi, top, bottom)  
 xi, yi = *self*.crossing(x\_prev, y\_prev, x, y, bottom)  
 *self*.bres\_int(xi, yi, x, y, *self*.black)  
 *self*.hor(xi, yi, x, y, top, bottom)  
 flag\_vis = flag\_t  
 x\_prev = x  
 y\_prev = y  
 *self*.obr\_r(x, y, x\_right, y\_right, top, bottom)

**Демонстрация работы программы.**

****

****