|  |
| --- |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА – Российский технологический университет"**  **РТУ МИРЭА** |

Институт кибербезопасности и цифровых технологий

Кафедра КБ-2 «Прикладные информационные технологии»

**ОТЧЕТ   
о выполнении контрольного домашнего задания №1.2**

по дисциплине**:  
«Формализованные модели и методы решения аналитических задач»**

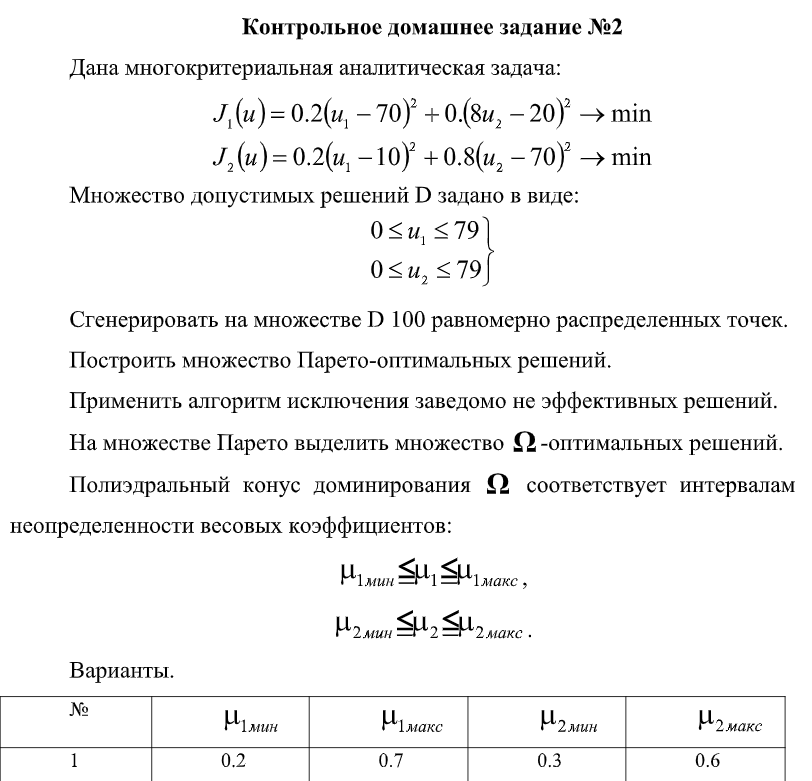
Выполнила: студентка 3 курса

Семёнова С.А. БИСО-03-20

Вариант 1

Проверил:*\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Москва 2023 г



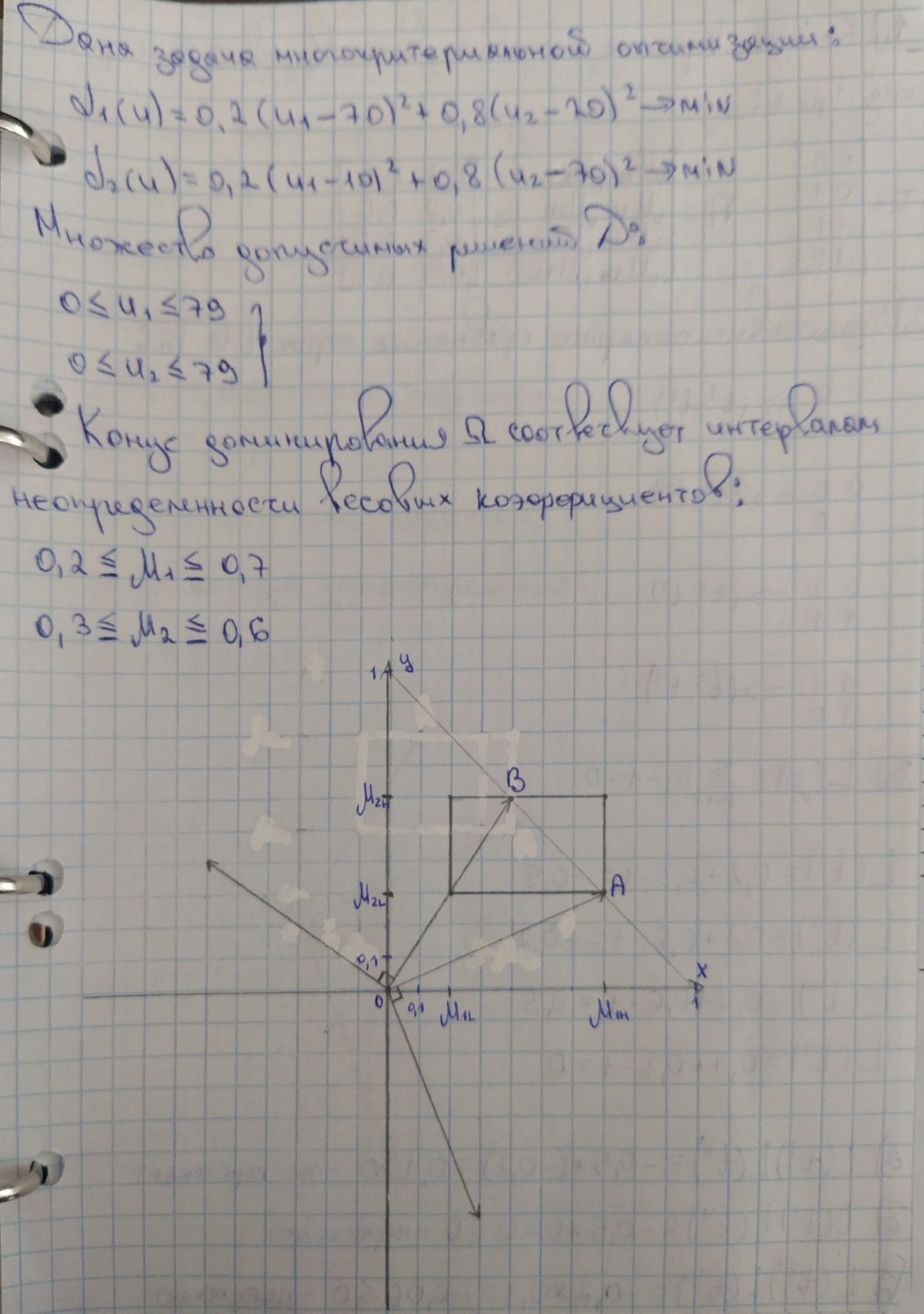
**Решение**

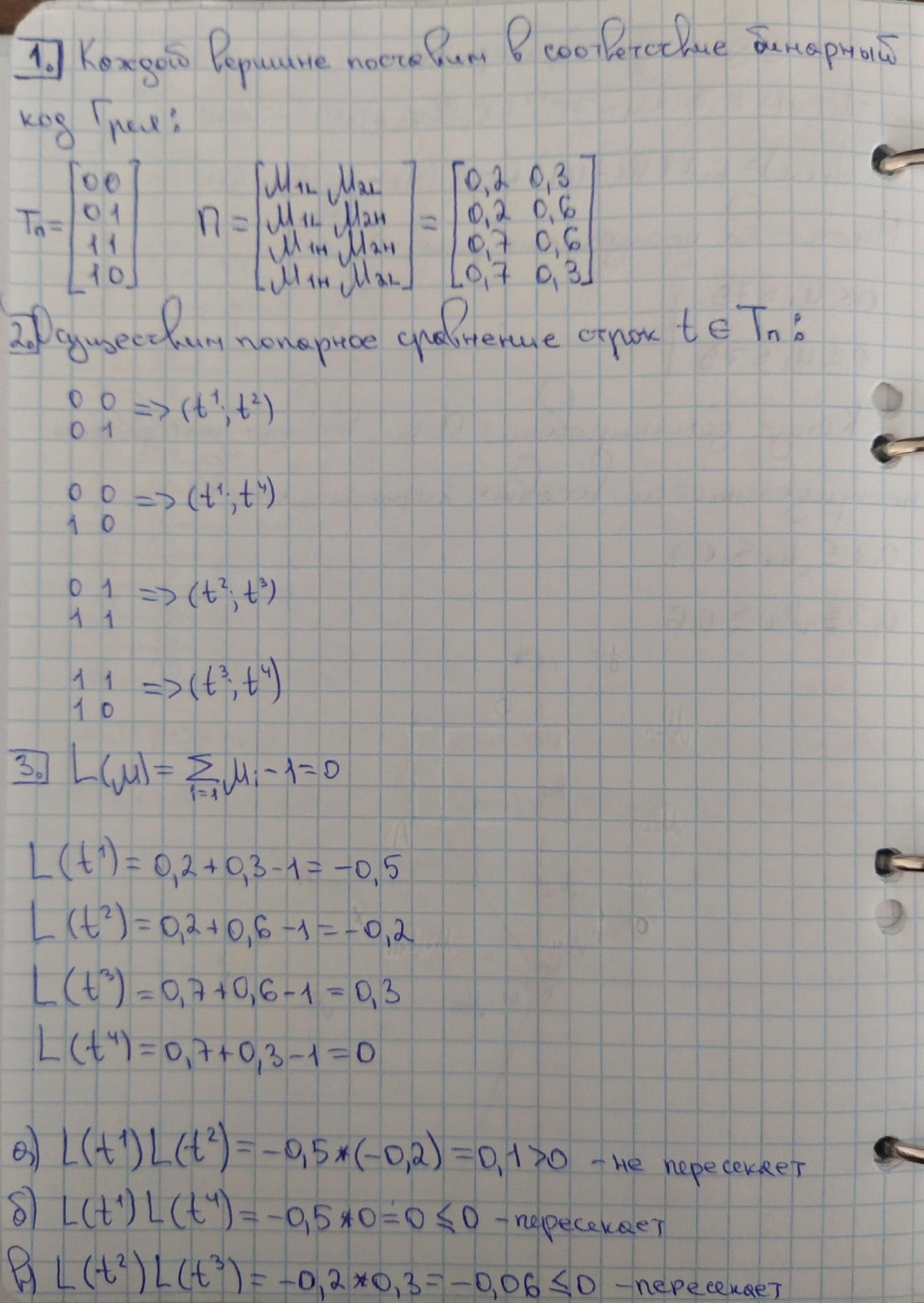
Создаю 100 точек, удовлетворяющих условию задачи, строю по

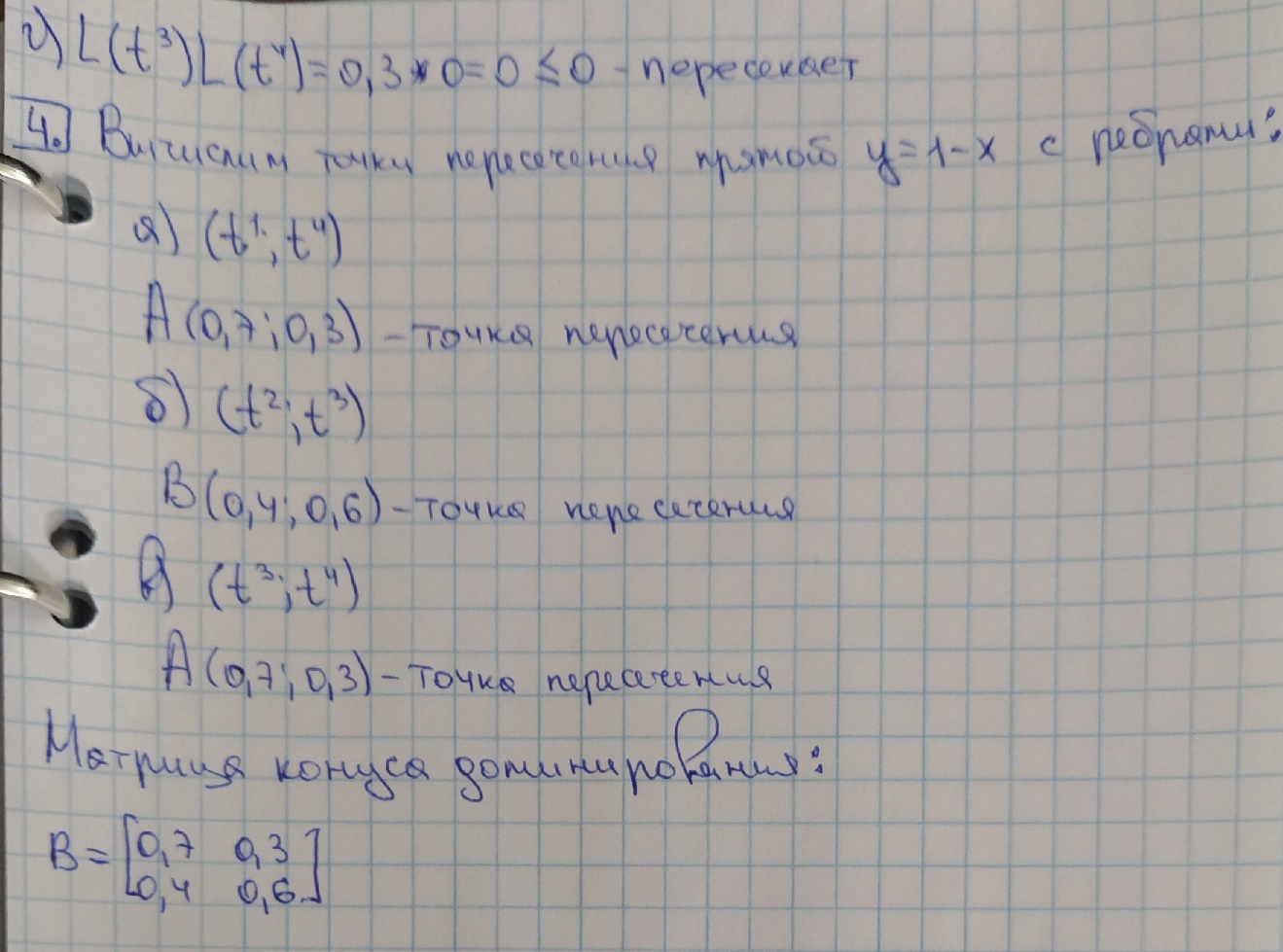
получившимся координатам таблицу значений u1 и u2.

Создаю программу на языке Python, отображающую данную

координатную плоскость с соответствующими результирующими наборами элементов по функциям J1 и J2.

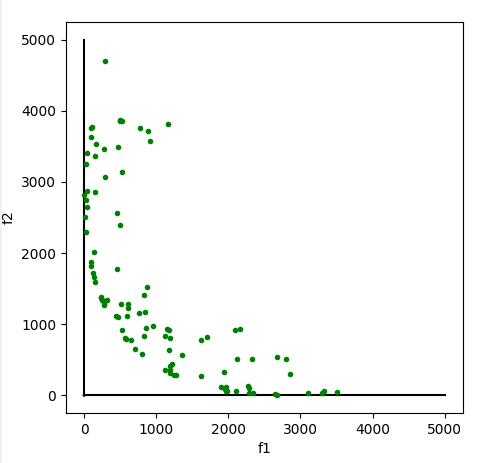




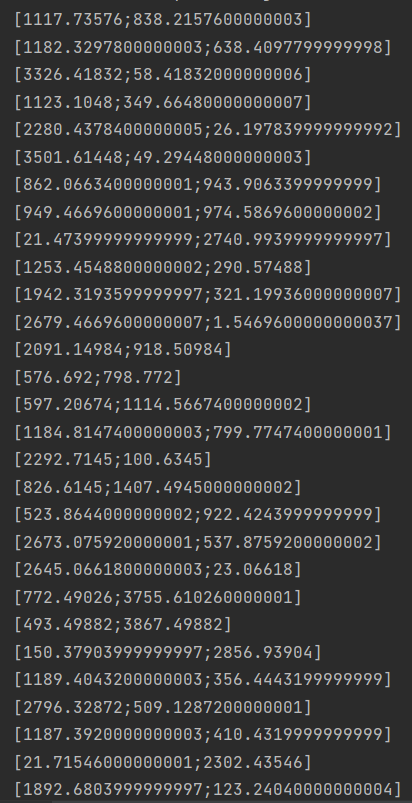


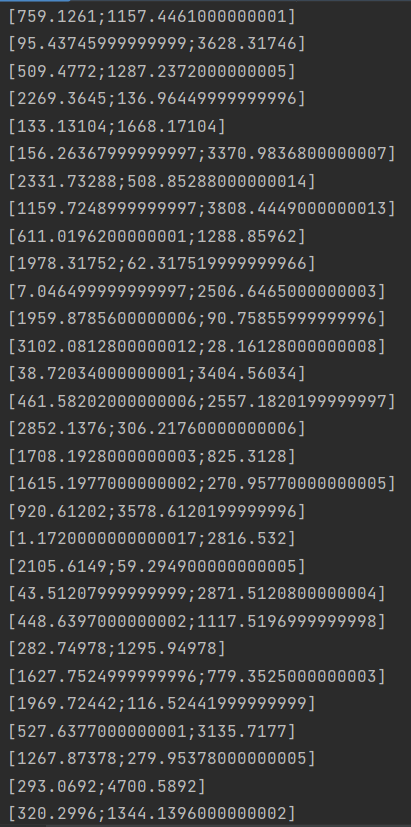
**Результат работы алгоритма:**

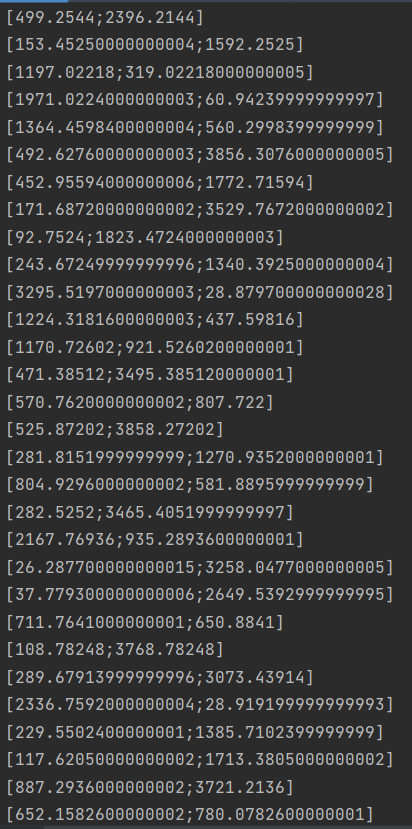
Графическое представление полученных точек.

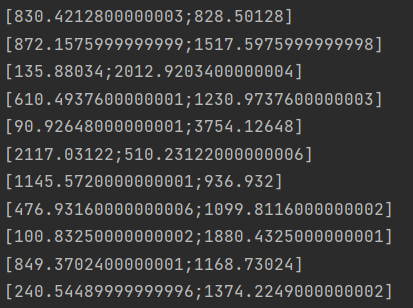


Координаты точек:

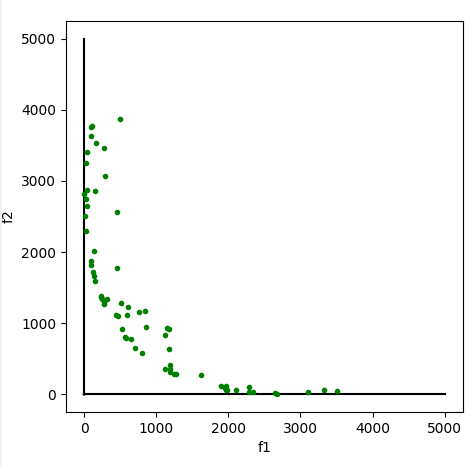






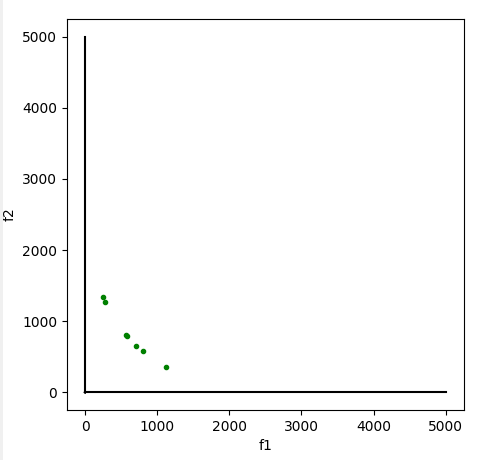


Строим множество Парето-оптимальных решений и применяем алгоритм исключения заведомо неэффективных решений:



На множестве Парето выделяем множество -оптимальных

решений:

****

**Листинг программы:**

import random  
from tkinter import \*  
from tkinter import messagebox  
  
import matplotlib.pyplot as plt  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
  
  
class Point(object):  
 def \_\_init\_\_(self, X, Y):  
 self.X = X  
 self.Y = Y  
 self.point = "+"  
  
  
N = 100  
  
list\_points = []  
list\_points2 = []  
window = Tk()  
window.geometry()  
window.configure()  
window.title("КДЗ\_№2")  
frame\_1 = Frame()  
frame\_1.configure()  
frame\_1.pack(side=BOTTOM)  
frame\_2 = Frame()  
frame\_2.configure()  
frame\_2.pack(side=TOP)  
frame\_3 = Frame(frame\_2)  
frame\_3.configure()  
frame\_3.pack(side=LEFT)  
frame\_4 = Frame(frame\_2)  
frame\_4.configure()  
frame\_4.pack(side=RIGHT)  
fig, ax = plt.subplots(figsize=(5, 5))  
canvas\_1 = FigureCanvasTkAgg(fig, master=frame\_3)  
canvas\_1.get\_tk\_widget().configure()  
  
  
def calculateABCD(nu\_x\_beg, nu\_x\_end, nu\_y\_beg, nu\_y\_end):  
 l = []  
 if nu\_y\_end >= -nu\_x\_end + 1 and nu\_y\_beg <= -nu\_x\_beg + 1:  
 if nu\_y\_end <= -nu\_x\_beg + 1:  
 if nu\_y\_beg <= -nu\_x\_end + 1:  
 xbeg = 1 - nu\_y\_end  
 xend = nu\_x\_end  
 ybeg = -nu\_x\_end + 1  
 yend = nu\_y\_end  
  
 else:  
 xbeg = 1 - nu\_y\_end  
 xend = 1 - nu\_y\_beg  
 ybeg = nu\_y\_beg  
 yend = nu\_y\_end  
 else:  
 if nu\_y\_beg >= -nu\_x\_beg + 1:  
 xbeg = nu\_x\_beg  
 xend = 1 - nu\_y\_beg  
 ybeg = nu\_y\_beg  
 yend = -nu\_x\_beg + 1  
 else:  
 xbeg = nu\_x\_beg  
 xend = nu\_x\_end  
 ybeg = 1 - nu\_x\_end  
 yend = 1 - nu\_x\_beg  
 l.append(xbeg)  
 l.append(xend)  
 l.append(ybeg)  
 l.append(yend)  
 return l  
  
  
def paint():  
 ax.clear()  
 plt.plot([0, 0], [0, 5000], color="k", ms=1)  
 plt.plot([2, 2.125], [0.125, 0], color="k", ms=1)  
 plt.plot([2, 2.125], [-0.125, 0], color="k", ms=1)  
 plt.plot([0, 5000], [0, 0], color="k", ms=1)  
 plt.plot([0, 0.125], [2.125, 2], color="k", ms=1)  
 plt.plot([0, -0.125], [2.125, 2], color="k", ms=1)  
 ax.set\_facecolor('white')  
 ax.set\_xlabel('f1')  
 ax.set\_ylabel('f2')  
  
  
def generate\_points():  
 paint()  
 list\_points.clear()  
  
 while len(list\_points) != N:  
 check = 0  
 X0 = 0.01 \* random.randint(0, 7900)  
 Y0 = 0.01 \* random.randint(0, 7900)  
 list\_points.append(  
 Point((0.2 \* (X0 - 70) \*\* 2 + 0.8 \* (Y0 - 20) \*\* 2), (0.2 \* (X0 - 10) \*\* 2 + 0.8 \* (Y0 - 70) \*\* 2)))  
 for i in range(len(list\_points)):  
 plt.plot(list\_points[i].X, list\_points[i].Y, "o", ms=3, color="g")  
 canvas\_1.draw()  
  
  
def Parettooptimization():  
 for i in range(len(list\_points)):  
 list\_points[i].dels = "+"  
 list\_points[i].seen = "-"  
 if len(list\_points) == 0:  
 messagebox.showinfo("Предупреждение:", "Cначала сгенерируйте точки")  
 else:  
 paint()  
 for i in range(len(list\_points)):  
 if list\_points[i].point != "-":  
 for k in range(len(list\_points)):  
 if k != i:  
 if list\_points[k].X >= list\_points[i].X and list\_points[k].Y >= list\_points[i].Y and \  
 list\_points[k].point == "+":  
 list\_points[k].point = "-"  
 break  
 if list\_points[i].point == "+":  
 plt.plot(list\_points[i].X, list\_points[i].Y, "o", ms=3, color="g")  
  
 canvas\_1.draw()  
  
  
def Ooptimization():  
 if list\_points2:  
 list\_points2.clear()  
 l = calculateABCD(0.2, 0.7, 0.3, 0.6) # POINTS  
 if len(list\_points) == 0:  
 messagebox.showinfo("Предупреждение:", "Cначала сгенерируйте точки")  
 else:  
 paint()  
 for i in range(len(list\_points)):  
 if list\_points[i].point != "-":  
 for k in range(len(list\_points)):  
 if k != i and list\_points[k].point != "-":  
 if list\_points[k].X <= list\_points[i].X and list\_points[k].Y <= list\_points[i].Y:  
 list\_points[k].point = "-"  
 for i in range(len(list\_points)):  
 list\_points2.append(Point(list\_points[i].X, list\_points[i].Y))  
 for i in range(len(list\_points2)):  
 if list\_points2[i].point != "-":  
 for k in range(len(list\_points2)):  
 if k != i:  
 if ((list\_points2[i].X - list\_points2[k].X) \* l[0] + (  
 list\_points2[i].Y - list\_points2[k].Y) \* l[1]) > 0 and (  
 (list\_points2[i].X - list\_points2[k].X) \* l[3] + (  
 list\_points2[i].Y - list\_points2[k].Y) \* l[2]) > 0:  
 list\_points2[i].point = "-"  
 for i in range(len(list\_points2)):  
 if list\_points2[i].point == "+":  
 plt.plot(list\_points2[i].X, list\_points2[i].Y, "o", ms=3, color="g")  
  
 canvas\_1.draw()  
  
  
def view\_all():  
 for p in list\_points:  
 print("[" + str(p.X) + ";" + str(p.Y) + "]")  
 for i in range(len(list\_points)):  
 list\_points[i].dels = "+"  
 list\_points[i].seen = "-"  
 if len(list\_points) == 0:  
 messagebox.showinfo("Предупреждение:", "Cначала сгенерируйте точки")  
 else:  
 paint()  
  
 for i in range(len(list\_points)):  
 plt.plot(list\_points[i].X, list\_points[i].Y, "o", ms=3, color="g")  
 canvas\_1.draw()  
  
  
generate\_btn = Button(frame\_1, text="Генерация точек", command=generate\_points, font="Times 10", width=50)  
generate\_btn.pack(side=TOP, fill=X, ipadx=6, padx=4, ipady=4, pady=5)  
optimize\_btn = Button(frame\_1, text="Паретто-оптимальные решения", command=Parettooptimization, font="Times 10")  
optimize\_btn.pack(side=TOP, fill=X, ipadx=6, padx=4, ipady=4, pady=0, )  
optimize2\_btn = Button(frame\_1, text="Ω-оптимльные решения", command=Ooptimization, font="Times 10")  
optimize2\_btn.pack(side=TOP, fill=X, ipadx=6, padx=4, ipady=4, pady=0, )  
showall\_btn = Button(frame\_1, text="Все", command=view\_all, font="Times 10")  
showall\_btn.pack(side=TOP, fill=X, ipadx=6, padx=4, ipady=4, pady=5)  
canvas\_1.get\_tk\_widget().pack(side=TOP, ipadx=6, padx=4, ipady=4, pady=5)  
paint()  
window.resizable(width=False, height=False)  
canvas\_1.draw()  
window.mainloop()

