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 *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине:

Разработка САПР

Тема лабораторной работы **Реализация геометрического решателя**

Студенты: **Абидоков Р. Ш., РК6-11М**
Пузаков Г. К., РК6-12М
Преподаватель: **Жук Д. М.**

Москва, 2020 г.

Оглавление

Задание на лабораторную работу

Реализовать графический редактор с возможностью создания точек и отрезков, наложения геометрических ограничений:

- Совпадения двух точек
- Расстояния между 2 точками
- Параллельности двух прямых
- Перпендикулярности двух прямых
- Угла между двумя прямыми
- Горизонтальности прямой
- Вертикальности прямой
- Принадлежности точки прямой

И возможностью расчета положения геометрии с учетом удовлетворения заданным ограничениям на основе принципа наименьших изменений.

Теоретическая часть

С математической точки зрения удовлетворение геометрическим ограничениям означает нахождение такого изменения координат точек $\Delta x_i, \Delta y_i$ (т.к. отрезок также описывается двумя точками), при котором удовлетворяется система уравнений

$$\begin{cases} \varphi_1(\Delta x_1, \Delta y_1, \dots, \Delta x_n, \Delta y_n) = 0 \\ \dots \\ \varphi_m(\Delta x_1, \Delta y_1, \dots, \Delta x_n, \Delta y_n) = 0 \end{cases}$$

являющаяся выражением наложенных ограничений, где n – количество точек, m – количество ограничений. Данная система уравнений в общем случае недоопределена, поскольку количество уравнений может быть меньше, чем 2n неизвестных.

Для того, чтобы сделать возможным нахождение однозначного решения, используется принцип наименьших изменений. Дополнительно к наложенным геометрическим ограничениям добавляется условие минимизации функции

$$\psi(\Delta x_1, \Delta y_1, \dots, \Delta x_n, \Delta y_n) = \frac{1}{2}\left(\sum_{i=1}^{n} \Delta x_i^2 + \sum_{i=1}^{n} \Delta y_i^2\right)$$

Таким образом исходная задача приводится к задаче оптимизации с ограничениями вида равенств и решается помощью метода множителей Лагранжа. Составляется функция Лагранжа вида:

$$F(\Delta x_1, \Delta y_1, \dots, \Delta x_n, \Delta y_n) = \psi(\dots) + \sum_{j=1}^{m} \lambda_j \varphi_j(\dots)$$

И разрешающая система уравнений

$$\begin{cases} \dfrac{\partial F(\dots)}{\partial \Delta x_i} = 0, & i = 1 \dots n \\ \dfrac{\partial F(\dots)}{\partial \Delta y_i} = 0, & i = 1 \dots n \\ \dfrac{\partial F(\dots)}{\partial \lambda_j} = 0, & j = 1 \dots m \end{cases}$$

Таким образом, количество уравнений совпадает с количеством неизвестных – и тех, и других 2n+m штук. С учетом вида функций ψ и φ:

$$\frac{\partial F(\dots)}{\partial \Delta x_i} = \frac{\partial \psi(\dots)}{\partial \Delta x_i} + \sum_{j=1}^{m} \lambda_j \frac{\varphi_j(\dots)}{\partial \Delta x_i} = \Delta x_i + \sum_{j=1}^{m} \lambda_j \frac{\varphi_j(\dots)}{\partial \Delta x_i}$$

$$\frac{\partial F(\dots)}{\partial \Delta y_i} = \frac{\partial \psi(\dots)}{\partial \Delta y_i} + \sum_{j=1}^{m} \lambda_j \frac{\varphi_j(\dots)}{\partial \Delta y_i} = \Delta y_i + \sum_{j=1}^{m} \lambda_j \frac{\varphi_j(\dots)}{\partial \Delta x_i}$$

$$\frac{\partial F(\dots)}{\partial \lambda_j} = \varphi_j(\dots)$$

Получаем итоговую систему:

$$\begin{cases} \Delta x_i + \sum_{j=1}^{m} \lambda_j \dfrac{\varphi_j(\dots)}{\partial \Delta x_i} = 0, & i = 1 \dots n \\ \Delta y_i + \sum_{j=1}^{m} \lambda_j \dfrac{\varphi_j(\dots)}{\partial \Delta y_i} = 0, & i = 1 \dots n \\ \varphi_j(\dots) = 0, & j = 1 \dots m \end{cases}$$

Выразим в явном виде функции ограничений φ_j (производные не приводятся для компактности).

- Расстояние между точками, и, как частный случай, их совпадение:

$$\varphi_{1-2}(\dots) = (x_1 + \Delta x_1 - x_2 - \Delta x_2)^2 + (y_1 + \Delta y_1 - y_2 - \Delta y_2)^2 - d^2$$

- Параллельность двух прямых (нулевое векторное произведение):

$$\varphi_3(\dots) = (x_1 + \Delta x_1 - x_2 - \Delta x_2)(y_3 + \Delta y_3 - y_4 - \Delta y_4) -$$
$$-(x_3 + \Delta x_3 - x_4 - \Delta x_4)(y_1 + \Delta y_1 - y_2 - \Delta y_2)$$

- Перпендикулярность двух прямых (нулевое скалярное произведение):

$$\varphi_4(\dots) = (x_1 + \Delta x_1 - x_2 - \Delta x_2)(x_3 + \Delta x_3 - x_4 - \Delta x_4) +$$
$$+(y_3 + \Delta y_3 - y_4 - \Delta y_4)(y_1 + \Delta y_1 - y_2 - \Delta y_2)$$

- Угол между двумя прямыми

$$\varphi_5(\dots) = (x_1 + \Delta x_1 - x_2 - \Delta x_2)(x_3 + \Delta x_3 - x_4 - \Delta x_4) +$$
$$+(y_3 + \Delta y_3 - y_4 - \Delta y_4)(y_1 + \Delta y_1 - y_2 - \Delta y_2) -$$
$$-\sqrt{(x_1 + \Delta x_1 - x_2 - \Delta x_2)^2 + (y_1 + \Delta y_1 - y_2 - \Delta y_2)^2} *$$
$$* \sqrt{(x_3 + \Delta x_3 - x_4 - \Delta x_4)^2 + (y_3 + \Delta y_3 - y_4 - \Delta y_4)^2} * cos\alpha$$

- Горизонтальность прямой

$$\varphi_6(\dots) = y_1 + \Delta y_1 - y_2 - \Delta y_2$$

- Вертикальность прямой

$$\varphi_7(\dots) = x_1 + \Delta x_1 - x_2 - \Delta x_2$$

- Принадлежность точки прямой

$$\varphi_8(\dots) = (x_1 + \Delta x_1 - x_p - \Delta x_p)(y_p + \Delta y_p - y_2 - \Delta y_2) -$$
$$-(x_p + \Delta x_p - x_2 - \Delta x_2)(y_1 + \Delta y_1 - y_p - \Delta y_p)$$

Интерфейс программы

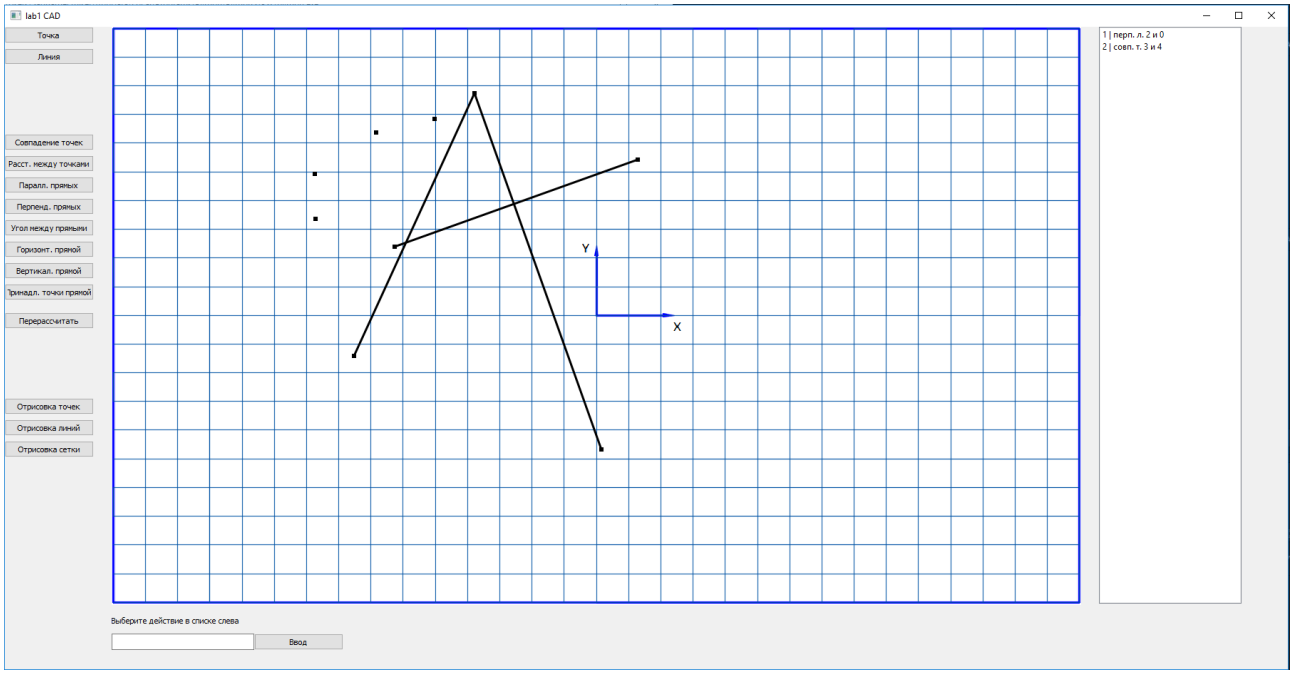

Рис. 1

Окно программы можно разделить на четыре зоны. В центре находится поле, внутри которого происходит построение графики. Справа находится лист, в котором выведен список наложенных геометрических ограничений.

Слева расположена панель с кнопками. Две верхних – "точка" и "линия" – отвечают, соответственно, за построение точки и линии. Ниже расположены кнопки наложения геометрических ограничений, под ними кнопка "перерасчитать", производящая повторное решение системы уравнений. Три отдельных кнопки ниже отвечают за включение-выключение отрисовки точек, линий и сетки поля.

Внизу текстовая строка, выводящая сообщения, и поле ввода с кнопкой, применяемое для ограничений расстояния между точками и угла между прямыми.

Примеры наложения ограничений

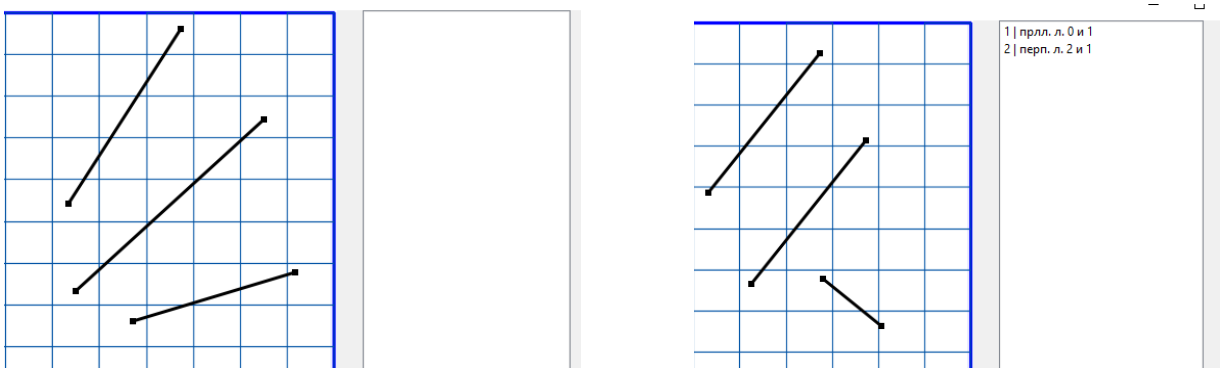

Рис. 2 Параллельность и перпендикулярность

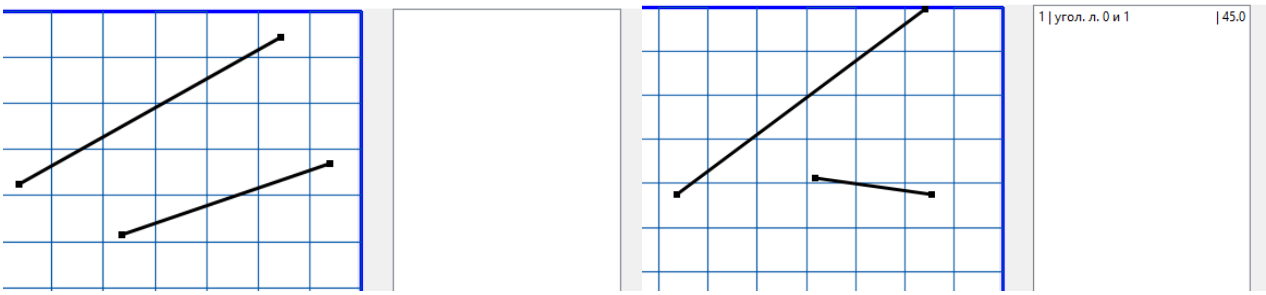

Рис. 3 Угол между прямыми

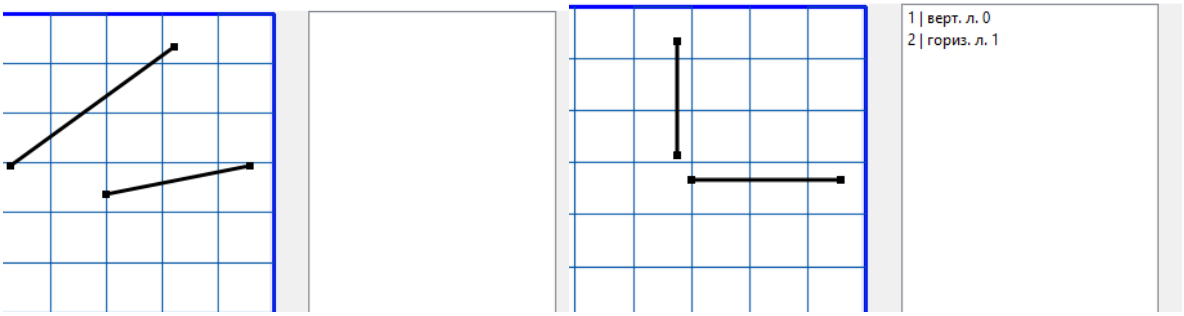

Рис. 4 Горизонтальность и вертикальность прямой

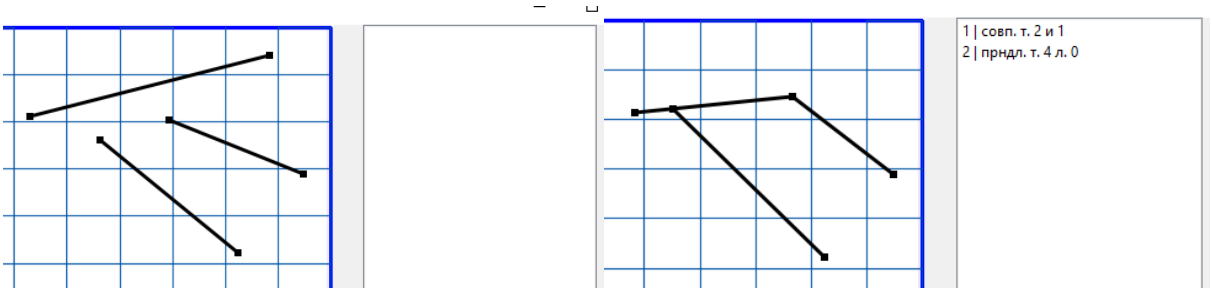

Рис. 5 Совпадения точек и принадлежность точки прямой

Описание программной реализации

Программа написана на языке Python 3.7.2 с использованием библиотеки PyQt5, отвечающей за графическую составляющую, и библиотек Numpy и Scipy для удобной работы с массивами и решения системы нелинейных уравнений.

С точки зрения архитектуры произведено разделение на три класса. Класс GuiClass непосредственно вызывается в main и обеспечивает отрисовку интерфейса, обработку событий, таких как щелчки мыши или нажатия кнопок, а также взаимодействие с классами StateClass и GeometryClass, экземпляры которых создаются в качестве полей.

StateClass отвечает за последовательную корректную смену состояний программы и вывод соответствующих сообщений пользователю, GeometryClass – за хранение координат точек, геометрических ограничений, формирование системы нелинейных уравнений и ее решение.

Удовлетворение граничным условием производится посредством вызова функции satisfy_constraints, приведенной в Листинге 1.

Листинг 1

```
def satisfy_constraints(self):
    n_points = self.points.shape[0]
    n_constr = self.constraints_idxs.shape[0]
    r_vec0 = np.zeros(n_points*2 + n_constr)
    r_vec = fsolve(self.equations_func, r_vec0)
    delta = np.reshape(r_vec[:n_points*2], [n_points, 2])
    self.points += delta
```

Искомый вектор r_vec размерности 2n+m имеет следующую структуру: первые 2n элементов – $\Delta x_i, \Delta y_i$ – т.е. изменения координат точек, оставшиеся m – неизвестные множители Лагранжа λ_j.

Формируется вектор начальных приближений r_vec0 (нулевых) и вместе с функцией equations_func, реализующей систему уравнений (т.е. по предложенному r_vec формирующей выходной вектор) передается функции fsolve модуля scipy.optimize, которая непосредственно ищет корни системы. Далее координаты точек изменяются на найденные величины (т.е. на первые 2n элемента r_vec).

Сама функция equations_func работает следующим образом: осуществляется проход по таблице, содержащей ограничения в виде столбцов "тип-индекс1-индекс2-величина1-величина2", где в зависимости от типа индексы могут соответствовать номерам точек либо линий, для каждого

ограничения в выходном векторе увеличивается значения производных по тем изменениям координат, которые участвуют в этом ограничении, а также производной по соответствующему множителю Лагранжа. Часть кода функции (для ограничений расстояния между точками и параллельности прямых) приведена в Листинге 2.

Полный исходный код программы находится в приложенных файлах и не включен в отчет в силу его громоздкости (порядка 1200 строк).

Листинг 2

```
137     def equations_func(self, l_vec):
138         # Кол-во уравнений = Ndx + Ndy + Nlambd = 2*Npoints + Nconstr
139         n_points = self.points.shape[0]
140         n_constr = self.constraints_idxs.shape[0]
141         n_eqs = n_points * 2 + n_constr
142         r_vec = np.zeros(n_eqs)
143         # Первыми идут производные по изменению координат
144         # Они равны изменениям координат + то, что приплюсуем дальше
145         r_vec[: n_points*2] = l_vec[: n_points*2]
146         # Дальше для каждого ограничения отдельные уравнения
147         for i, constr_i in enumerate(self.constraints_idxs):
148             type, idx_k, idx_l = constr_i
149             val1, val2 = self.constraints_values[i]
150             if (type == 1 or type == 2):  # Расстояние между точками
151                 # Производная по первому иксу
152                 r_vec[2*idx_k] += self.Dfi2Ddxk(l_vec, idx_k, idx_l, val1, i)
153                 # Производная по первому игреку
154                 r_vec[2*idx_k+1] += self.Dfi2Ddyk(l_vec, idx_k, idx_l, val1, i)
155                 # Производная по второму иксу
156                 r_vec[2*idx_l] += self.Dfi2Ddxl(l_vec, idx_k, idx_l, val1, i)
157                 # Производная по второму игреку
158                 r_vec[2*idx_l+1] += self.Dfi2Ddyl(l_vec, idx_k, idx_l, val1, i)
159                 # производная по лямбде
160                 r_vec[2*n_points+i] = self.fi2(l_vec, idx_k, idx_l, val1)
161             elif (type == 3):               # Параллелность линий
162                 idx_kp, idx_lp = self.lines[idx_k]
163                 idx_pp, idx_qp = self.lines[idx_l]
164                 r_vec[2*idx_kp] += self.Dfi3Ddxk(l_vec, idx_kp, idx_lp, idx_pp, idx_qp, i)
165                 r_vec[2*idx_kp+1] += self.Dfi3Ddyk(l_vec, idx_kp, idx_lp, idx_pp, idx_qp, i)
166                 r_vec[2*idx_lp] += self.Dfi3Ddxl(l_vec, idx_kp, idx_lp, idx_pp, idx_qp, i)
167                 r_vec[2*idx_lp+1] += self.Dfi3Ddyl(l_vec, idx_kp, idx_lp, idx_pp, idx_qp, i)
168                 r_vec[2*idx_pp] += self.Dfi3Ddxp(l_vec, idx_kp, idx_lp, idx_pp, idx_qp, i)
169                 r_vec[2*idx_pp+1] += self.Dfi3Ddyp(l_vec, idx_kp, idx_lp, idx_pp, idx_qp, i)
170                 r_vec[2*idx_qp] += self.Dfi3Ddxq(l_vec, idx_kp, idx_lp, idx_pp, idx_qp, i)
171                 r_vec[2*idx_qp+1] += self.Dfi3Ddyq(l_vec, idx_kp, idx_lp, idx_pp, idx_qp, i)
172                 r_vec[2*n_points+i] = self.fi3(l_vec, idx_kp, idx_lp, idx_pp, idx_qp)
```

Присутствующие недочеты

1. Отсутствует возможность изменения масштаба и движения системы координат
2. После расчета нового положения точек, часть из них может оказаться вне поля, ограниченного рамкой. В этом случае графическое отображение будет не совсем корректным (Рис. 6)

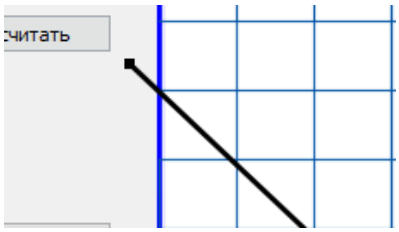

Рис. 6 (выход линии за границу рамки)

3. Если выбрать ограничение в списке справа и таким образом активировать подсветку точек/линий, а затем отключить отображение соответствующих элементов, то подсвеченная геометрия продолжит отображаться

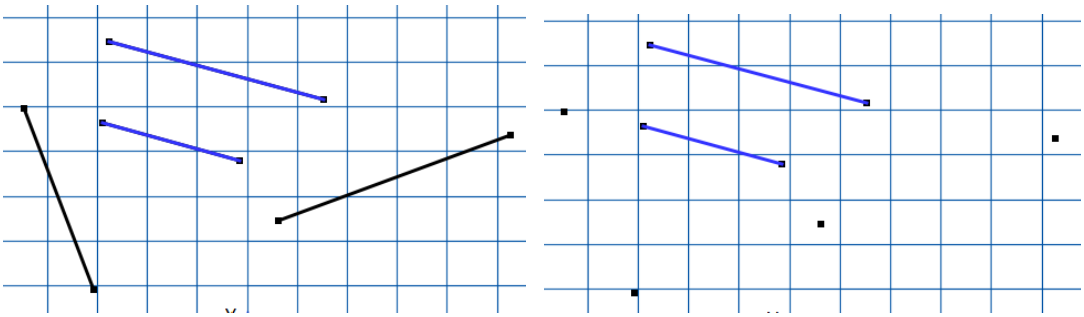

Рис. 7 (при выключении отображения линий)

4. При наложении ограничения на расстояние между точками, отличное от нулевого, поведение может быть необычным – например, в случае, когда оптимальным решением являлось бы сближение точек вдоль линии, их соединяющей, может произойти поворот, как это показано на Рис. 8. Возможным объяснением такого поведения могут служить проблемы решателя при поиске корней уравнения.

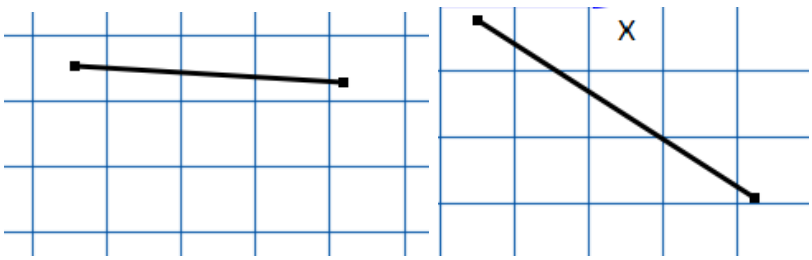

Рис. 8

Заключение

Был реализован графический редактор с возможностью построения двухмерной геометрии, состоящей из точек и отрезков и наложения геометрических ограничений. При реализации использован язык программирования Python, графическая библиотека PyQt5 и математические библиотеки Numpy, Pandas. Суммарный объем исходного кода составил примерно 1200 строк.

Для указанных выше недочетов могут быть предложены следующие решения:

1. *Масштаб и движение системы координат* – разделение системы координат, использующейся при отрисовке, и абсолютной системы координат, в которой записаны положения точек, непосредственный пересчет одной в другую при отрисовке с помощью матриц перехода
2. *Выход части объектов за границу поля* – выполнять проверку на принадлежность полю, не отрисовывать объекты, целиком находящие вне поля, для линии, пересекающей рамку, конечную точку заменять на точку пересечения линии и границы
3. *Продолжение подсветки элементов, на которые наложены ограничения* – добавить проверку флага при отрисовке, аналогичную той, что присутствует для неподсвеченных линий
4. *Поворот при наложении ограничения расстояния* – может помочь смена решателя и/или использование начальных приближений, отличных от нулевых.