Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Кафедра информатики и прикладной математики

Лабораторная работа №4
Дисциплина «Символьные вычисления»

Вариант 19

Выполнил:
Съестов Дмитрий Вячеславович
Группа P3317

Преподаватель:
Кореньков Юрий Дмитриевич

Санкт-Петербург
2019

Задание

Цель - изучить методы формирования динамических визуализаций на основе символьных представлений данных и зависимостей между ними.
Разработанную в ходе выполнения предыдущего задания программу дополнить командой построения визуализации в символьной форме в виде входного формата, при выполнении которой на экран выводится графическое представление результатов вычисления аргументов, переданных ей в соответствии с вариантом. Размеры графического окна не фиксированные, при манипуляциях с ним визуализация должна обновляться.

График

```
class PlotWindow(QMainWindow):
    def __init__(self, data):
        super().__init__()
        self.initializeUI(data)

    def initializeUI(self, data):
        self.setWindowTitle('График')
        self.setGeometry(0, 0, 500, 400)
        self.setMinimumSize(400, 320)
        self.center()
        self.setFocus()

        self._main = QWidget()
        self.setCentralWidget(self._main)
        layout = QVBoxLayout(self._main)
        self.setLayout(layout)

        self.figure = Figure()
        self.canvas = FigureCanvas(self.figure)
        layout.addWidget(self.canvas)
        self.plot(data)

    def center(self):
        screen_center = QDesktopWidget().availableGeometry().center()
        frame = self.frameGeometry()
        frame.moveCenter(screen_center)
        self.move(frame.topLeft())

    def plot(self, data):
        for xs, ys, label in data:
            ax = self.figure.add_subplot(111)
            ax.plot(xs, ys, label=label)
            ax.legend()
        self.canvas.draw()
```

```
def createPlot(data):
    app = QApplication(sys.argv)
    window = PlotWindow(data)
    window.show()
    app.exec()
```

Функция Julia

```
function plot(args)
    n = length(args)
    if (n < 3 || n % 3 != 0)
        error("plot() requires a number of arguments divisible by 3")
    end

    plots = []
    data = [args[i:i+2] for i in 1:3:n]
    for (expr, a, b) in data
        a, b = eval(a), eval(b)
        xs, ys = @eval Context begin
            f = x -> $expr
            xs = collect($a:0.1:$b)
            ys = map(f, xs)
            xs, ys
        end
        push!(plots, (xs, ys, "y = $expr"))
    end

    Main.plotwindow.createPlot(plots)
end
```

Для отображения графиков используется программа на PyQt, принимающая массив кортежей вида **(expr, a, b)**, где **expr** – выражение из x в y, **a** и **b** – границы по x. Используется шаг 0.1. Для рисования графика используется matplotlib. График размещён в компоновщике, поэтому при изменении размеров окна он перерисовывается.