Министерство науки и высшего образования РФ

Санкт-Петербургский государственный политехнический университет

Петра Великого

Институт компьютерных наук и технологий

Высшая школа киберфизических систем и управления

**ПОРТФОЛИО**

по дисциплине «Идентификация систем»

Выполнила:

Студент гр. 3532704/80501 <> О.А. Хвещук

Проверил:

Доцент <> В. А. Онуфриев

Санкт-Петербург 2021 г.

**Содержание**

[Задание №1 4](#_Toc51620038)

[Постановка задачи 4](#_Toc51620039)

[Решение задачи 4](#_Toc51620040)

[Алгоритмы 4](#_Toc51620041)

[Код 6](#_Toc51620042)

[Примеры работы программы 8](#_Toc51620043)

[Задание №2 8](#_Toc51620044)

[Постановка задачи 8](#_Toc51620045)

[Решение задачи 9](#_Toc51620046)

[Алгоритмы 11](#_Toc51620047)

[Код 14](#_Toc51620048)

[Пример работы программы 17](#_Toc51620049)

[Задание №3 18](#_Toc51620050)

[Постановка задачи 18](#_Toc51620051)

[Решение задачи 19](#_Toc51620052)

[Алгоритмы 19](#_Toc51620053)

[Код 19](#_Toc51620054)

[Пример работы программы 23](#_Toc51620055)

[Задание №4 24](#_Toc51620056)

[Постановка задачи 24](#_Toc51620057)

[Решение задачи 25](#_Toc51620058)

[Алгоритмы 25](#_Toc51620059)

[Код 30](#_Toc51620060)

[Пример работы программы 41](#_Toc51620061)

[Задание №5 42](#_Toc51620062)

[Постановка задачи 42](#_Toc51620063)

[Решение задачи 42](#_Toc51620064)

[Алгоритмы 43](#_Toc51620065)

[Код 43](#_Toc51620066)

[Пример работы программы 52](#_Toc51620067)

[Задание №6 54](#_Toc51620068)

[Постановка задачи 54](#_Toc51620069)

[Решение задачи 54](#_Toc51620070)

[Код 55](#_Toc51620071)

[Пример работы программы 56](#_Toc51620072)

[Задание №7 57](#_Toc51620073)

[Постановка задачи 57](#_Toc51620074)

[Решение задачи 57](#_Toc51620075)

[Код 58](#_Toc51620076)

[Пример работы программы 62](#_Toc51620077)

[Приложение 63](#_Toc51620078)

# Задание №1

## Постановка задачи

Задана импульсная переходная характеристика системы в виде функции k(t). Необходимо построить график данной функции, и получить передаточную функцию, соответствующую данной импульсной переходной характеристике.

## Решение задачи

Связь передаточной функции и импульсной переходной характеристики выглядит следующим образом:

K(p)=W(p)\*L(δ(t))

K(p)=W(p)\*1=

k(t)=

## Алгоритмы

**update\_graph –** Функция для расчета точек и вывода их на график

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Вид |
| fx | Заданное K(t) | string | Входной |
| t | Время, переменная предназначена для считывания функции fx | String, float | Промежуточный |
| X1 | Массив точек по t | float | Выходной |
| n | Длина X1 | int | Промежуточный |
| y | Массив точек по ординате K(t) | float | Выходной |



конец

начало

Считать fx с поля

Создать массив x1 от 0 до 100 с шагом. n=длине массива x1

Создать пустой массив y длинной n

t = x1[i]

y[i] = fx(t)

i от 0 до n



Fx считывается с поля и мат. выражение?

Добавить график x1,y c осями t и K(t)

Очистить поле и вывести сообщение:

«Введите корректное выражение K(t)»

**on\_click –** Функция для расчета W(p) по введенному K(t)

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Вид |
| p | Комплексная переменная, предназначена для преобразования Лапласа | string | Промежуточный |
| t | Время, переменная предназначена для считывания функции fx | String | Промежуточный |
| fx | K(t) | String | Входной |
| y | Передаточная K(p) | String | Выходной |



конец

начало

Вывести y в поле



Fx считывается с поля и мат. выражение?

Очистить поле и вывести сообщение:

«Введите корректное выражение K(t)»

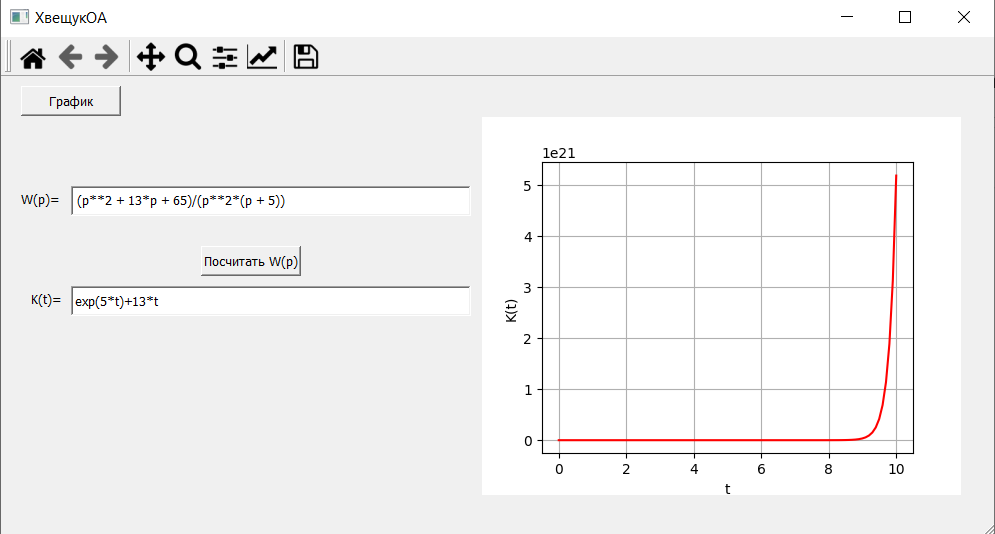
update\_graph

## Код

Main.py:

**import** sys  
**import** matplotlib  
matplotlib.use(**'Qt5Agg'**)  
  
**from** PyQt5 **import** QtCore, QtWidgets  
**from** PyQt5.QtWidgets **import** \*  
**from** matplotlib.backends.backend\_qt5agg **import** FigureCanvasQTAgg, NavigationToolbar2QT **as** NavigationToolbar  
**from** matplotlib.figure **import** Figure  
  
**from** mplwidget **import** MplWidget  
  
**import** pandas **as** pd  
**import** numpy **as** np  
**from** sympy **import** \*  
  
**class** MplCanvas(FigureCanvasQTAgg):  
  
 **def** \_\_init\_\_(self, parent=**None**, width=5, height=4, dpi=100):  
 fig = Figure(figsize=(width, height), dpi=dpi)  
 self.axes = fig.add\_subplot(111)  
 super(MplCanvas, self).\_\_init\_\_(fig)  
  
  
**class** MainWindow(QtWidgets.QMainWindow):  
  
 **def** \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(MainWindow, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.initUI()  
  
 **def** initUI(self):  
 *# Create a placeholder widget to hold our toolbar and canvas.* self.setObjectName(**"MainWindow"**)  
 self.resize(993, 501)  
 self.centralwidget = QtWidgets.QWidget(self)  
 self.centralwidget.setObjectName(**"centralwidget"**)  
 self.MplWidget = MplWidget(self.centralwidget)  
 self.MplWidget.setGeometry(QtCore.QRect(470, 30, 501, 400))  
 self.MplWidget.setMinimumSize(QtCore.QSize(480, 320))  
 self.MplWidget.setObjectName(**"MplWidget"**)  
 self.setCentralWidget(self.centralwidget)  
 self.addToolBar(NavigationToolbar(self.MplWidget.canvas, self))  
 *# вывод результата* self.rez = QTextEdit(self)  
 self.rez.move(70, 150)  
 self.rez.resize(400, 30)  
 *# добавление надписи* self.lbl = QLabel(self)  
 self.lbl.move(20, 155)  
 self.lbl.setText(**'W(p)='**)  
 self.lbl.adjustSize()  
 *# добавлене кнопок* btn = QPushButton(**"График"**, self)  
 btn.move(20, 50)  
 *# если нажали на кнопку выполнить функции* btn.clicked.connect(self.update\_graph)  
 self.statusBar()  
 self.setWindowTitle(**'ХвещукОА'**)  
  
 self.textbox = QLineEdit(self)  
 self.textbox.move(70, 250)  
 self.textbox.resize(400, 30)  
 self.lbl1 = QLabel(self)  
 self.lbl1.move(30, 255)  
 self.lbl1.setText(**'K(t)='**)  
 self.lbl1.adjustSize()  
  
 self.btnWp = QPushButton(**'Посчитать W(p)'**, self)  
 self.btnWp.move(200, 210)  
 self.btnWp.clicked.connect(self.on\_click)  
 self.show()  
 **def** update\_graph(self):  
 t = symbols(**'t'**)  
 **try**:  
 fx = self.textbox.text()  
 g = eval(fx)  
 **except**:  
 self.rez.clear()  
 QMessageBox.critical(self, **"Ошибка "**, **"Введите корректное выражение K(t)"**, QMessageBox.Ok)  
 **return** x1 = np.linspace(0, 10, 100)  
 n = len(x1)  
 y = [0] \* n  
 **for** i **in** range(n):  
 t = x1[i]  
 y[i] = eval(fx)  
 self.MplWidget.canvas.axes.clear()  
 self.MplWidget.canvas.axes.plot(x1, y, **'r'**)  
 self.MplWidget.canvas.axes.set\_xlabel(**'t'**)  
 self.MplWidget.canvas.axes.set\_ylabel(**'K(t)'**)  
 self.MplWidget.canvas.axes.grid()  
 self.MplWidget.canvas.draw()  
 **def** on\_click(self):  
 p = symbols(**'p'**)  
 t = symbols(**'t'**)  
 **try**:  
 fx = self.textbox.text()  
 g = eval(fx)  
 **except**:  
 self.rez.clear()  
 QMessageBox.critical(self, **"Ошибка "**, **"Введите корректное выражение K(t)"**, QMessageBox.Ok)  
 **return** y = laplace\_transform(g, t, p, noconds=**True**)*#считает прямое преобразование лапласа* self.rez.setText(str(y))  
 self.update\_graph  
app = QtWidgets.QApplication(sys.argv)  
w = MainWindow()  
app.exec\_()

## Примеры работы программы



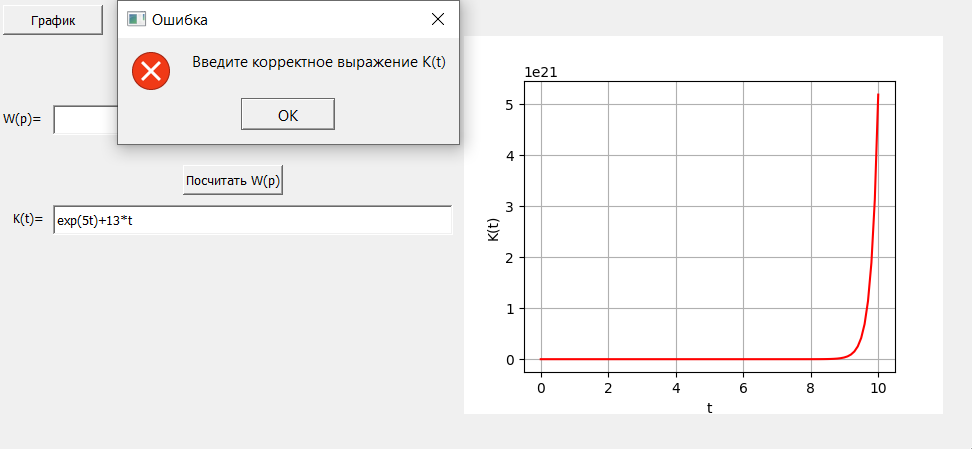


Рисунок 1 – Примеры работы программы

# Задание №2

## Постановка задачи

Заданы желаемые параметры системы:

1. Перерегулирование: 1-5%;
2. Время установления: 4 с;
3. Установившееся значение: 2,6.

Необходимо получить передаточную функцию, переходная характеристика которой будет соответствовать заданным параметрам.

## Решение задачи

Для того чтобы решить эту задачу было принято решение использовать вещественный интерполяционный метод, который позволяет по имеющейся переходной характеристике получить передаточную функцию, соответствующую ей. Связь между передаточной функцией и переходной характеристикой выглядит следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

Обратимся к формуле прямого преобразования Лапласа:

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

В нее входят функция изображение F(p), преобразуемая функция f(t) и комплеская переменная . Модели в виде F(p) обладают преимуществом при использовании операций интегрирования и дифференцирования. Так, достаточно сложной операции дифференцирования функции времени f(t) соответсвует в области изображений в случае нулевых начальных условий значительно более простая операция умножения функции на переменную p: . Интегрированию в области оригиналов соответсвует деление функции-изображения F(p) на переменную p. Источником трудностей при таких операциях является мнимая составляющая в комплеской переменной .

При комплексная переменная p вырождается в вещественную , а исходная формула (5) принимает вид:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

Условия существования и единственности функции определяются сходимостью интеграла (6). Поэтому накладывается ограничение на переменную :

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

В приложении к линейным САУ, когда функция f(t) представляет собой временную динамическую характеристику, сходимость обеспечивают выбором соответсвующего значения параметра C. Так, для устойчивой системы, имеющий импульсную переходную характеристику , можно принять C = 0. В случае использования переходной характеристики , соответвующей устойчивой системе, условие сходимости интеграла определяется тоже достаточно просто или C > 0.

Из всего выше сказанного можно получить следующие выражения для передаточных функций:

|  |  |  |
| --- | --- | --- |
|  |  | (8) |
|  |  | (9) |

Или, если отсутствует аналитические выражения для k(t) и h(t), но имеются точки:

|  |  |  |
| --- | --- | --- |
|  |  | (10) |
|  |  | (11) |

Для того чтобы воспользоваться данным методом необходимо представить передаточную функцию в следующем виде:

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

Значение левой части данного выражения получим из формулы (11). Далее составим систему линейных алгебраических уравнений, из которой найдем неизвестные коэффициенты. Количество уравнений будет равно .

В начале работы были подобраны точки, соответвующие заданным параметрам переходной характеристики и записаны в файл. Ниже представлена часть программы, который считывает эти точки из файла, записывает в массив и выводит на график.

Полученная передаточная функция выводится в интерфейс программы, а далее происходит обратный процесс – получение переходной характеристики системы по ее передаточной функции. Для этого воспользуемся численным обращением преобразования Лапласа:

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

Используя теорему вычетов, получим:

|  |  |  |
| --- | --- | --- |
|  |  | (14) |

После всех математических преобразований получим:

|  |  |  |
| --- | --- | --- |
|  |  | (15) |

где d1 и d2 вещественная и мнимая, соответственно, части комплексно-сопряженных корней.

## Алгоритмы

**Kar –** Функция вычисляет коэффициенты передаточной с помощью решения СЛАУ

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Вид |
| d | Массив из дельт | float | Входной |
| x | Точки по оси абсцисс | float | Входной |
| y | Точки по оси ординат | float | Входной |
| Matr | Матрица системы | float | Промежуточный |
| b | Вектор свободных членов | float | Промежуточный |
| K | Массив коэффициентов передаточной | float | Выходной |



конец

начало

K=0

i от 0 до длины d

|K|=K

Matr += [[d[i], 1, -W \* d[i] \*\* 2, -W \* d[i]]]

b += [W]

W=PF(x, y, d[i])

K = np.linalg.solve(Matr, b)

Встроенная функция для нахождения решений СЛАУ методом Гаусса

**PF –** Функция для расчета W(p)

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Вид |
| d | Дельта | float | Входной |
| x | Точки по оси абсцисс | float | Входной |
| y | Точки по оси ординат | float | Входной |
| W | Значение передаточной | float | Выходной |

конец

начало

i от 0 до длины x-1

W = W\*d

W=0

W += y[i+1] \* np.exp(-d \* x[i+1]) \* (x[i + 1] - x[i])

**OBG –**поиск переходной характеристики и получение передаточной

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Вид |
| d | Массив из дельт | float | Входной |
| x | Точки по оси абсцисс | float | Входной |
| y | Точки по оси ординат | float | Входной |
| K | Массив коэффициентов передаточной | float | Промежуточный |
| p | Комплексная переменная, предназначена для преобразования Лапласа | string | Промежуточный |
| t | Точка по оси абсцисс | float | Промежуточный |
| X1 | Точки по оси абсцисс, заданные по последнему значению x | float | Выходной |
| h | точки переходной характеристики | float | Выходной |
| fp | Передаточная | String  function | Выходной |



конец

начало

i от 0 до длины d

t = x1[i]

h[i]=

K = Kar(d, x, y)

K=0

x1, h,fp(p)=0

Fp(p)= (K[0] \* p + K[1]) / (K[2] \* p \*\* 2 + K[3] \* p + 1)/p

x1 = Массив от 0.01 до x[-1] длиной 100

**ParamH –** Функция для расчета параметров переходной характеристики

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Вид |
| t | Точки по оси абсцисс | float | Входной |
| h | Точки по оси ординат | float | Входной |
| top | Верхняя граница установившегося значения | float | Промежуточный |
| bot | Верхняя граница установившегося значения | float | Промежуточный |
| hmax | Максимальное значение h | float | Промежуточный |
| i | Счетчик | int | Промежуточный |
| treg | Время регулирования | float | Выходной |
| hyst | Установившееся значение | float | Выходной |
| reg | Регулирование | float | Выходной |

конец

начало

treg = t[i]

hyst = h[-1]

hmax = max(h)

reg = (hmax - hyst) / hyst \* 100

top = hyst \* 1.05

bot = hyst \* 0.95

i = длина h – 1

i -= 1

Пока h[i] > bot и h[i] <= top

## Код

ForMatr.py

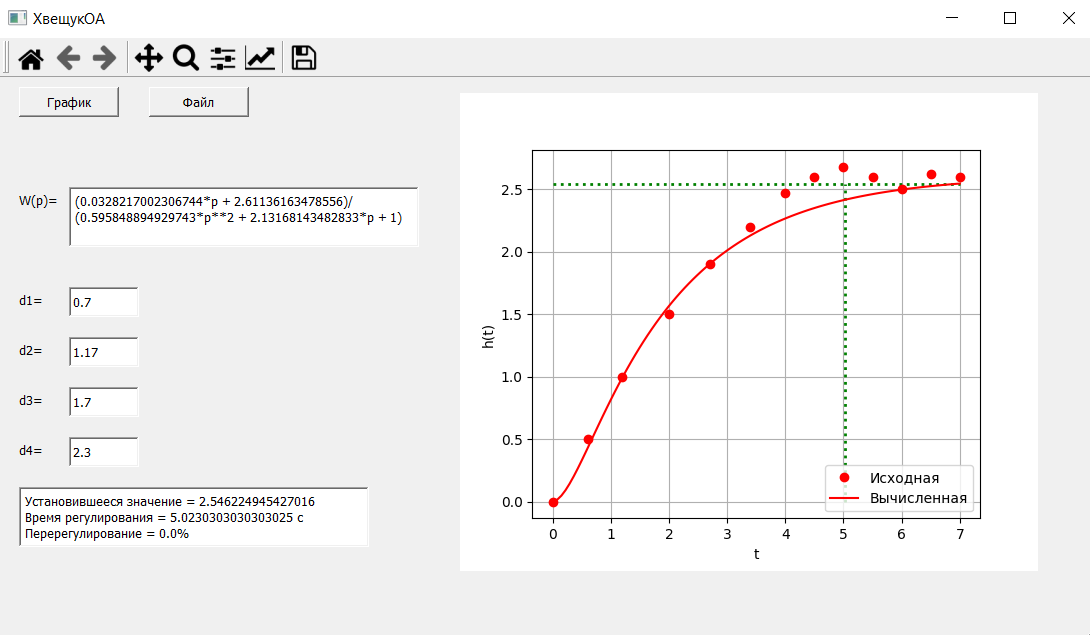
**from** sympy **import** \*  
**from** mpmath **import** \*  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**def** PF(x, y, d): *#передаточная функция по точкам* W = 0  
 **for** i **in** range(len(x) - 1):  
 W += y[i+1] \* np.exp(-d \* x[i+1]) \* (x[i + 1] - x[i])  
 W = W\*d  
 **return** (W)  
**def** dmax(x,y):*#метод половинного аргумента для поиска дельты (не используется в конечном коде, но в некоторых случаях крайне полезная)* k=0.2\*y[-1]  
 a=0.01  
 b=3  
 **while** abs(b-a)/2>10E-16:  
 c = (a + b) / 2  
 **if** ((PF(x,y,a)-k)\*(PF(x,y,c)-k)) >= 0:  
 a=c  
 **else**:  
 b=c  
 **return** c  
**def** W(K, d):*#передаточная с коэффициентами* W = (K[0] \* d + K[1]) / (K[2] \* d \*\* 2 + K[3] \* d + 1)  
 **return** (W)  
**def** OBG(d, x, y):*#поиск переходной характеристики и получение передаточной* K = Kar(d, x, y)  
 **if** K.all()==0:  
 **return** 0,0,0  
 p = symbols(**'p'**,positive=true)  
 *# y= integrate (g,(x,0,oo))  
# H = str(re(inverse\_laplace\_transform(W(K, p)/ p, p, t)))* fp=**lambda** p: (K[0] \* p + K[1]) / (K[2] \* p \*\* 2 + K[3] \* p + 1)/p  
 x1 = np.linspace(0.01, x[-1], 100)  
 h = [0] \* len(x1)  
 **for** i **in** range(len(x1)):  
 t = x1[i]  
 h[i]=float(invertlaplace(fp, t, method=**'talbot'**))  
 **return** (x1, h,fp(p))  
**def** ParamH(t, h):  
 hyst = h[-1]  
 hmax = max(h)  
 reg = (hmax - hyst) / hyst \* 100  
 top = hyst \* 1.05  
 bot = hyst \* 0.95  
 i = len(h) - 1  
 **while** h[i] > bot **and** h[i] <= top:  
 i -= 1  
 treg = t[i]  
 **return** treg, hyst,reg  
def Kar(d, x, y): *#вычисление коэффициентов* Matr = []  
 b = []  
 for i in range(len(d)):  
 W=PF(x, y, d[i])  
 Matr += [[d[i], 1, -W \* d[i] \*\* 2, -W \* d[i]]]  
 b += [W]  
 K = np.linalg.solve(Matr, b)  
 if abs(K).all()!=K.all():  
 K=0  
 return K

main.py

**import** sys  
**import** matplotlib  
**from** PyQt5 **import** QtCore, QtWidgets  
**from** PyQt5.QtWidgets **import** \*  
**from** matplotlib.backends.backend\_qt5agg **import** FigureCanvasQTAgg, NavigationToolbar2QT **as** NavigationToolbar  
**from** matplotlib.figure **import** Figure  
**import** pandas **as** pd  
**import** numpy **as** np  
**from** mplwidget **import** MplWidget  
**from** ForMatr **import** \*  
  
**class** MplCanvas(FigureCanvasQTAgg):  
  
 **def** \_\_init\_\_(self, parent=**None**, width=5, height=4, dpi=100):  
 fig = Figure(figsize=(width, height), dpi=dpi)  
 self.axes = fig.add\_subplot(111)  
 super(MplCanvas, self).\_\_init\_\_(fig)  
  
  
**class** MainWindow(QtWidgets.QMainWindow):  
  
 **def** \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(MainWindow, self).\_\_init\_\_(\*args, \*\*kwargs)  
 sc = MplCanvas(self, width=5, height=4, dpi=100)  
 df.plot(ax=sc.axes)  
 self.initUI()  
 **def** initUI(self):  
 *# Create a placeholder widget to hold our toolbar and canvas.* self.setObjectName(**"MainWindow"**)  
 self.resize(1100, 600)  
 self.centralwidget = QtWidgets.QWidget(self)  
 self.centralwidget.setObjectName(**"centralwidget"**)  
 self.MplWidget = MplWidget(self.centralwidget)  
 self.MplWidget.setGeometry(QtCore.QRect(450, 5, 600, 500))  
 self.MplWidget.setMinimumSize(QtCore.QSize(450, 350))  
 self.MplWidget.setObjectName(**"MplWidget"**)  
 self.setCentralWidget(self.centralwidget)  
 self.addToolBar(NavigationToolbar(self.MplWidget.canvas, self))  
 self.rez = QTextEdit(self)  
 self.rez.move(70, 150)  
 self.rez.resize(350, 60)  
 *# добавление надписи* self.lbl = QLabel(self)  
 self.lbl.move(20, 155)  
 self.lbl.setText(**'W(p)='**)  
 self.lbl.adjustSize()  
 self.d1 = QLineEdit(self)  
 self.d1.move(70, 250)  
 self.d1.resize(70, 30)  
 self.d2 = QLineEdit(self)  
 self.d2.move(70, 300)  
 self.d2.resize(70, 30)  
 self.d3 = QLineEdit(self)  
 self.d3.move(70, 350)  
 self.d3.resize(70, 30)  
 self.d4 = QLineEdit(self)  
 self.d4.move(70, 400)  
 self.d4.resize(70, 30)  
 self.ld1 = QLabel(self)  
 self.ld1.move(20, 255)  
 self.ld1.setText(**'d1='**)  
 self.ld1.adjustSize()  
 self.ld2 = QLabel(self)  
 self.ld2.move(20, 305)  
 self.ld2.setText(**'d2='**)  
 self.ld2.adjustSize()  
 self.ld3 = QLabel(self)  
 self.ld3.move(20, 355)  
 self.ld3.setText(**'d3='**)  
 self.ld3.adjustSize()  
 self.ld4 = QLabel(self)  
 self.ld4.move(20, 405)  
 self.ld4.setText(**'d4='**)  
 self.ld4.adjustSize()  
 self.lotv = QTextEdit(self)  
 self.lotv.move(20, 450)  
 self.lotv.resize(350, 60)  
 *# добавлене кнопок* btn1 = QPushButton(**"График"**, self)  
 btn1.move(20, 50)  
 btn2 = QPushButton(**"Файл"**, self)  
 btn2.move(150, 50)  
 **global** filename  
 filename = **'AB'** *# если нажали на кнопку выполнить функции* btn1.clicked.connect(self.update\_graph)  
 btn2.clicked.connect(self.getFileName)  
 self.statusBar()  
 self.setWindowTitle(**'ХвещукОА'**)  
 self.show()  
 **def** update\_graph(self):  
 A = np.loadtxt(filename)  
 x = A[:, 0]  
 y = A[:, 1]  
 **try**:  
 d1 = self.d1.text()  
 d2 = self.d2.text()  
 d3 = self.d3.text()  
 d4 = self.d4.text()  
 **if** d1 == **"" or** d2 == **"" or** d3 == **"" or** d4 == **""**:  
 dmin = -np.log(0.25 / 2.6) / 4  
 d1 = dmin  
 d2 = dmin \* 2  
 d3 = dmin \* 3  
 d4 = dmin \* 4  
 self.d1.setText(str(d1))  
 self.d2.setText(str(d2))  
 self.d3.setText(str(d3))  
 self.d4.setText(str(d4))  
 **else**:  
 d1 = float(d1)  
 d2 = float(d2)  
 d3 = float(d3)  
 d4 = float(d4)  
 **except**:  
 self.rez.clear()  
 QMessageBox.critical(self, **"Ошибка "**, **"Введите корректные дельты"**, QMessageBox.Ok)  
 **return** d = [d1,d2,d3,d4]  
 x1,h,W=OBG(d,x,y)  
 **if** h==0:  
 self.rez.clear()  
 QMessageBox.critical(self, **"Ошибка "**, **"При таких дельтах W(p) имеет отрицательные коэффициенты"**, QMessageBox.Ok)  
 **return** treg, hyst, reg=ParamH(x1, h)  
 self.lotv.clear()  
 otv = **'Установившееся значение = '** + str(hyst) + **'\nВремя регулирования = '** + str(treg) + **' c\n'** + **'Перерегулирование = '** + str(reg)+**'%'** self.lotv.setText(otv)  
 self.MplWidget.canvas.axes.clear()  
 self.MplWidget.canvas.axes.plot(x,y,**'ro'**)  
 self.MplWidget.canvas.axes.plot(x1, h,**'r'**)  
 self.MplWidget.canvas.axes.hlines(hyst, 0, x[-1], color=**'g'**, linewidth=2, linestyle=**':'**)  
 self.MplWidget.canvas.axes.vlines(treg, 0, hyst, color=**'g'**, linewidth=2, linestyle=**':'**)  
 self.MplWidget.canvas.axes.legend((**'Исходная'**, **'Вычисленная'**), loc=**'lower right'**)  
 self.rez.setText(str(W))  
 self.MplWidget.canvas.axes.set\_xlabel(**'t'**)  
 self.MplWidget.canvas.axes.set\_ylabel(**'h(t)'**)  
 self.MplWidget.canvas.axes.grid()  
 self.MplWidget.canvas.draw()  
 **def** getFileName(self):  
 **global** filename  
 filename, filetype = QFileDialog.getOpenFileName(self, **"All Files(\*.\*)"**)  
app = QtWidgets.QApplication(sys.argv)  
w = MainWindow()  
app.exec\_()

## Пример работы программы

В интерфейсе программы есть поля для ввода узлов. Кнопка график позволяет расчитать передаточную, вывести график h(t) и получившиеся параметры h(t)



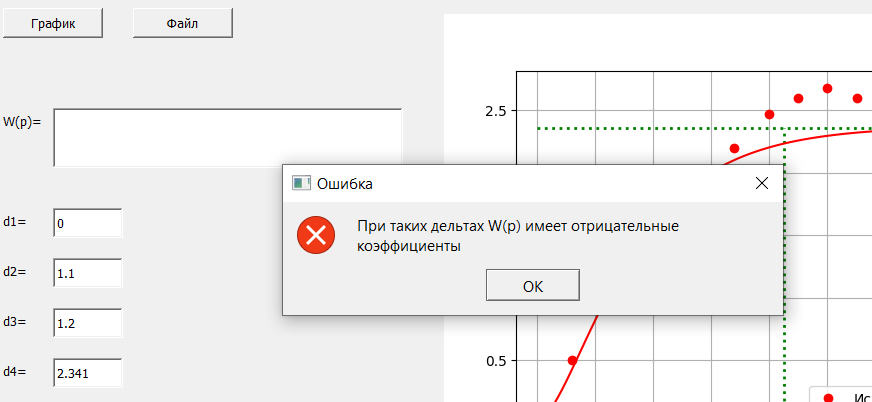


Рисунок 2 – Пример работы программы.

# Задание №3

## Постановка задачи

Задана импульсная переходная характеристика в виде точек функции k(t). Необходимо произвести идентификацию системы, т. е. получить соответствующие данной переходной характеристике передаточные функции в структуре 3/3. Выбор узлов должен производится вручную (например, «ползунками»). Отобразить фактическое перерегулирование и время регулирования.

## Решение задачи

Для того чтобы решить эту задачу было принято решение, как и в прошлом задании, использовать вещественный интерполяционный метод, который позволяет по имеющейся импульсной переходной характеристике получить передаточную функцию, соответствующую ей (см. формулу 8). Суть данного метода была описана в прошлом задании.

Из файла считываются точки, записываются в массив, выводятся на график. Далее узлы считываются с «ползунков». Составляется СЛАУ, которая решается методом Гаусса. После нахождения коэффициентов передаточный функции они выводятся в интерфейс программы и для нахождения переходной характеристики используется вещественное преобразование Лапласа, как описано в предыдущем задании.

## Алгоритмы

Алгоритмы аналогичны алгоритмам из 2 работы. Меняется лишь матрица системы Matr, и вид передаточной функции pf.

Matr+= [[d[i]\*\* 2, d[i]\*\* 1, 1,-W\* d[i] \*\* 2, -W \* d[i] \*\* 1,-W]],

Fp(p)=(K[0] \* p\*\*3 + K[1]\* p\*\*2+K[2] \* p+1) / (K[3] \* p\*\*3 + K[4]\*p\*\*2+K[5]\*p+1)/p. Также незначительно изменена функция PF(x, y, d), но принцип вычисления остался прежним.

## Код

ForMatr.py

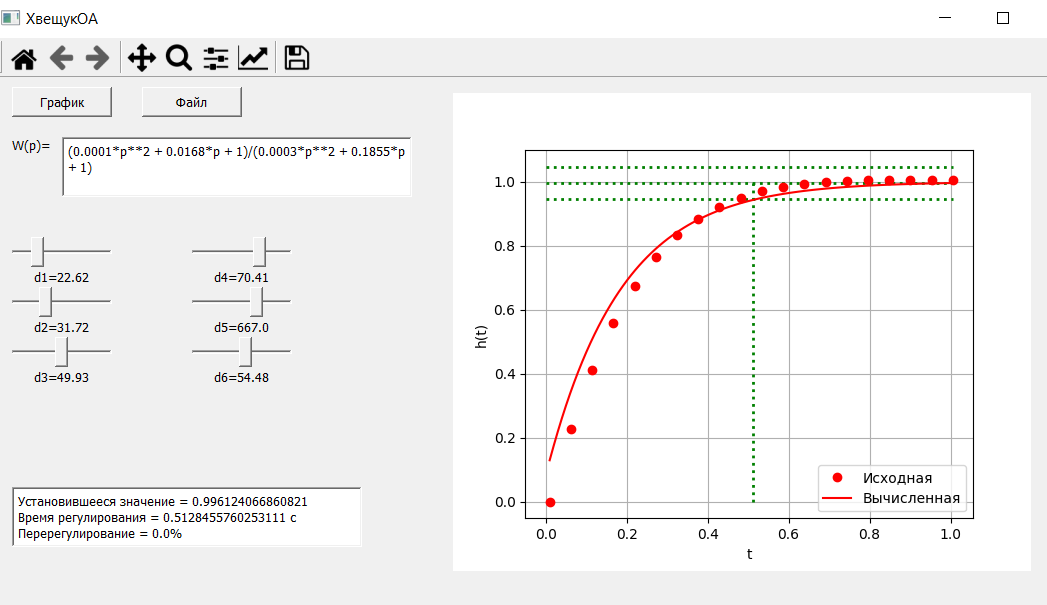
from sympy import \*  
from mpmath import \*  
import numpy as np  
def PF(x, y, d): *#передаточная функция по точкам* W = 0  
 for i in range(len(x)):  
 if (i == 0):  
 dt = x[i + 1] - x[i]  
 else:  
 dt = x[i] - x[i - 1]  
 W += y[i] \* np.exp(-d \* x[i]) \* dt  
 W = W\*d  
 return (W)  
def ParamH(t, h):  
 hyst = h[-1]  
 hmax = max(h)  
 reg = (hmax - hyst) / hyst \* 100  
 top = hyst \* 1.05  
 bot = hyst \* 0.95  
 i = len(h) - 1  
 while h[i] > bot and h[i] <= top:  
 i -= 1  
 treg = t[i]  
 return treg, hyst,reg  
def dmax(x,y):*#метод половинного аргумента для поиска дельты* k=0.1\*2.6  
*# a=0.000008* a=0.01  
 b=3  
 while abs(b-a)/2>10E-16:  
 c = (a + b) / 2  
 if ((PF(x,y,a)-k)\*(PF(x,y,c)-k)) >= 0:  
 a=c  
 else:  
 b=c  
 *#print(PF(x, y, c) - k,c)* return c  
def OBG(d, x, y):*#поиск переходной характеристики и получение передаточной* p = symbols(**'p'**,positive=true)  
 K = Kar(d, x, y)  
 K=np.around(K,decimals=4,out=None)  
 fp=lambda p: (K[0] \* p\*\*3 + K[1]\* p\*\*2+K[2] \* p+1) / (K[3] \* p\*\*3 + K[4]\* p\*\*2+K[5] \* p+1)/p  
 if not np.all(abs(K)==K):  
 return 0,0,fp(p)  
 x1 = np.linspace(0.01, y[-1], 100)  
 h = [0] \* len(x1)  
 for i in range(len(x1)):  
 t = x1[i]  
 h[i]=float(invertlaplace(fp, t, method=**'talbot'**))  
 return (x1,h,fp(p))  
def Kar(d, x, y): *#вычисление коэффициентов* Matr = []  
 b = []  
 for i in range (len(d)):  
 W=PF(x, y, d[i])  
 Matr += [[d[i]\*\* 2, d[i]\*\* 1, 1,-W\* d[i] \*\* 2, -W \* d[i] \*\* 1,-W]]  
 b += [(W-1)/d[i]]  
 K = np.linalg.solve(Matr, b)  
 return K

Main.py

**import** sys  
**import** matplotlib  
**from** PyQt5 **import** QtCore, QtWidgets  
**from** PyQt5.QtCore **import** Qt  
**from** PyQt5.QtWidgets **import** \*  
**from** matplotlib.backends.backend\_qt5agg **import** FigureCanvasQTAgg, NavigationToolbar2QT **as** NavigationToolbar  
**from** matplotlib.figure **import** Figure  
**import** pandas **as** pd  
**import** numpy **as** np  
**from** mplwidget **import** MplWidget  
**from** ForMatr **import** \*  
  
**class** MplCanvas(FigureCanvasQTAgg):  
  
 **def** \_\_init\_\_(self, parent=**None**, width=5, height=4, dpi=100):  
 fig = Figure(figsize=(width, height), dpi=dpi)  
 self.axes = fig.add\_subplot(111)  
 super(MplCanvas, self).\_\_init\_\_(fig)  
  
  
**class** MainWindow(QtWidgets.QMainWindow):  
  
 **def** \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(MainWindow, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.initUI()  
  
 **def** initUI(self):  
 *# Create a placeholder widget to hold our toolbar and canvas.* self.setObjectName(**"MainWindow"**)  
 self.resize(1100, 600)  
 self.centralwidget = QtWidgets.QWidget(self)  
 self.centralwidget.setObjectName(**"centralwidget"**)  
 self.MplWidget = MplWidget(self.centralwidget)  
 self.MplWidget.setGeometry(QtCore.QRect(450, 5, 600, 500))  
 self.MplWidget.setMinimumSize(QtCore.QSize(450, 350))  
 self.MplWidget.setObjectName(**"MplWidget"**)  
 self.setCentralWidget(self.centralwidget)  
 self.addToolBar(NavigationToolbar(self.MplWidget.canvas, self))  
 self.rez = QTextEdit(self)  
 self.rez.move(70, 100)  
 self.rez.resize(350, 60)  
 self.lbl = QLabel(self)  
 self.lbl.move(20, 100)  
 self.lbl.setText(**'W(p)='**)  
 self.lbl.adjustSize()  
 t= 10000  
 t1=5  
 r=100  
 self.sld1 = QSlider(Qt.Horizontal, self)  
 self.sld1.setRange(r, t)  
 self.sld1.setFocusPolicy(Qt.NoFocus)  
 self.sld1.setPageStep(t1)  
 self.sld1.move(20,200)  
 self.ld1 = QLabel(**'d1='**+str(r/100), self)  
 self.ld1.setAlignment(Qt.AlignCenter | Qt.AlignVCenter)  
 self.ld1.setMinimumWidth(30)  
 self.ld1.move(20, 225)  
 self.sld1.valueChanged.connect(self.updateLabel1)  
 self.sld2 = QSlider(Qt.Horizontal, self)  
 self.sld2.setRange(r, t)  
 self.sld2.setFocusPolicy(Qt.NoFocus)  
 self.sld2.setPageStep(t1)  
 self.sld2.move(20,250)  
 self.ld2 = QLabel(**'d2='**+str(r/100), self)  
 self.ld2.setAlignment(Qt.AlignCenter | Qt.AlignVCenter)  
 self.ld2.setMinimumWidth(30)  
 self.ld2.move(20, 275)  
 self.sld2.valueChanged.connect(self.updateLabel2)  
 self.sld3 = QSlider(Qt.Horizontal, self)  
 self.sld3.setRange(r, t)  
 self.sld3.setFocusPolicy(Qt.NoFocus)  
 self.sld3.setPageStep(t1)  
 self.sld3.move(20,300)  
 self.ld3 = QLabel(**'d3='**+str(r/100), self)  
 self.ld3.setAlignment(Qt.AlignCenter | Qt.AlignVCenter)  
 self.ld3.setMinimumWidth(30)  
 self.ld3.move(20, 325)  
 self.sld3.valueChanged.connect(self.updateLabel3)  
 self.sld4 = QSlider(Qt.Horizontal, self)  
 self.sld4.setRange(r, t)  
 self.sld4.setFocusPolicy(Qt.NoFocus)  
 self.sld4.setPageStep(t1)  
 self.sld4.move(200,200)  
 self.ld4 = QLabel(**'d4='**+str(r/100), self)  
 self.ld4.setAlignment(Qt.AlignCenter | Qt.AlignVCenter)  
 self.ld4.setMinimumWidth(30)  
 self.ld4.move(200,225)  
 self.sld4.valueChanged.connect(self.updateLabel4)  
 self.sld5 = QSlider(Qt.Horizontal, self)  
 self.sld5.setRange(r, 100000)  
 self.sld5.setFocusPolicy(Qt.NoFocus)  
 self.sld5.setPageStep(t1)  
 self.sld5.move(200,250)  
 self.ld5 = QLabel(**'d5='**+str(r/100), self)  
 self.ld5.setAlignment(Qt.AlignCenter | Qt.AlignVCenter)  
 self.ld5.setMinimumWidth(30)  
 self.ld5.move(200, 275)  
 self.sld5.valueChanged.connect(self.updateLabel5)  
 self.sld6 = QSlider(Qt.Horizontal, self)  
 self.sld6.setRange(r, t)  
 self.sld6.setFocusPolicy(Qt.NoFocus)  
 self.sld6.setPageStep(t1)  
 self.sld6.move(200,300)  
 self.ld6 = QLabel(**'d6='**+str(r/100), self)  
 self.ld6.setAlignment(Qt.AlignCenter | Qt.AlignVCenter)  
 self.ld6.setMinimumWidth(30)  
 self.ld6.move(200,325)  
 self.sld6.valueChanged.connect(self.updateLabel6)  
  
 self.lotv = QTextEdit(self)  
 self.lotv.move(20, 450)  
 self.lotv.resize(350, 60)  
 *# добавлене кнопок* btn1 = QPushButton(**"График"**, self)  
 btn1.move(20, 50)  
 btn2 = QPushButton(**"Файл"**, self)  
 btn2.move(150, 50)  
 **global** filename  
 filename = **'file8.csv'** fx=**'2\*exp(-x)'** *# если нажали на кнопку выполнить функции* btn1.clicked.connect(self.update\_graph2)  
 btn2.clicked.connect(self.getFileName)  
 self.statusBar()  
 self.setWindowTitle(**'ХвещукОА'**)  
 self.show()  
 **def** updateLabel1(self, value):self.ld1.setText(**'d1='**+str(value/100))  
 **def** updateLabel2(self, value): self.ld2.setText(**'d2='** + str(value / 100))  
 **def** updateLabel3(self, value): self.ld3.setText(**'d3='** + str(value / 100))  
 **def** updateLabel4(self, value): self.ld4.setText(**'d4='** + str(value / 100))  
 **def** updateLabel5(self, value): self.ld5.setText(**'d5='** + str(value / 100))  
 **def** updateLabel6(self, value): self.ld6.setText(**'d6='** + str(value / 100))  
 **def** updateLabel7(self, value):self.ld7.setText(**'d7='** + str(value / 100))  
  
 **def** update\_graph2(self):  
 **if** filename==**'file8.csv'**:  
 y=np.loadtxt(filename, delimiter=**','**)  
 y = y[:2000:100]  
 x = np.linspace(0.01, y[-1], len(y))  
 **else**:  
 y = np.loadtxt(filename, delimiter=**','**)  
 y = y[:len(y)-1:int((len(y)-1)/20)]  
 x = np.linspace(0.01, y[-1], len(y))  
 d1=self.ld1.text()  
 d1=float(d1[3:])  
 d2 = self.ld2.text()  
 d2 = float(d2[3:])  
 d3 = self.ld3.text()  
 d3 = float(d3[3:])  
 d4 = self.ld4.text()  
 d4 = float(d4[3:])  
 d5 = self.ld5.text()  
 d5 = float(d5[3:])  
 d6 = self.ld6.text()  
 d6 = float(d6[3:])  
 d=[d1,d2,d3,d4,d5,d6]  
 **if** len(d)!=len(set(d)):  
 QMessageBox.critical(self, **"Ошибка "**, **"Дельты не должны равняться друг другу"**, QMessageBox.Ok)  
 **return** x1,h,W=OBG(d,x,y)  
 **if** h == 0:  
 QMessageBox.critical(self, **"Ошибка "**, **"Коэффициенты меньше 0\nW(p)="** + str(W), QMessageBox.Ok)  
 self.lotv.clear()  
 self.rez.clear()  
 self.MplWidget.canvas.axes.clear()  
 **return** treg, hyst, reg = ParamH(x1, h)  
 hreg1 = 0.95 \* hyst  
 hreg2 = 1.05 \* hyst  
 self.lotv.clear()  
 otv = **'Установившееся значение = '** + str(hyst) + **'\nВремя регулирования = '** + str(  
 treg) + **' c\n'** + **'Перерегулирование = '** + str(reg) + **'%'** self.lotv.setText(otv)  
 self.MplWidget.canvas.axes.clear()  
 self.MplWidget.canvas.axes.plot(x, y, **'ro'**)  
 self.MplWidget.canvas.axes.plot(x1, h, **'r'**)  
 self.MplWidget.canvas.axes.hlines(hyst, 0, x[-1], color=**'g'**, linewidth=2, linestyle=**':'**)  
 self.MplWidget.canvas.axes.hlines(hreg1, 0, x[-1], color=**'g'**, linewidth=2, linestyle=**':'**)  
 self.MplWidget.canvas.axes.hlines(hreg2, 0, x[-1], color=**'g'**, linewidth=2, linestyle=**':'**)  
 self.MplWidget.canvas.axes.vlines(treg, 0, hyst, color=**'g'**, linewidth=2, linestyle=**':'**)  
 self.MplWidget.canvas.axes.legend((**'Исходная'**, **'Вычисленная'**), loc=**'lower right'**)  
 self.rez.setText(str(W))  
 self.MplWidget.canvas.axes.set\_xlabel(**'t'**)  
 self.MplWidget.canvas.axes.set\_ylabel(**'h(t)'**)  
 self.MplWidget.canvas.axes.grid()  
 self.MplWidget.canvas.draw()  
 **def** getFileName(self):  
 **global** filename  
 **try**:  
 filename, filetype = QFileDialog.getOpenFileName(self, **"All Files(\*.\*)"**)  
 **except**:  
 QMessageBox.critical(self, **"Ошибка "**, **"Неверное имя файла"**, QMessageBox.Ok)  
 *# print (os.path.basename(filename))*app = QtWidgets.QApplication(sys.argv)  
w = MainWindow()  
app.exec\_()

## Пример работы программы

В интерфейсе программы есть «ползунки» для ввода узлов. Кнопка график позволяет расчитать передаточную, вывести график h(t) и получившиеся параметры h(t)



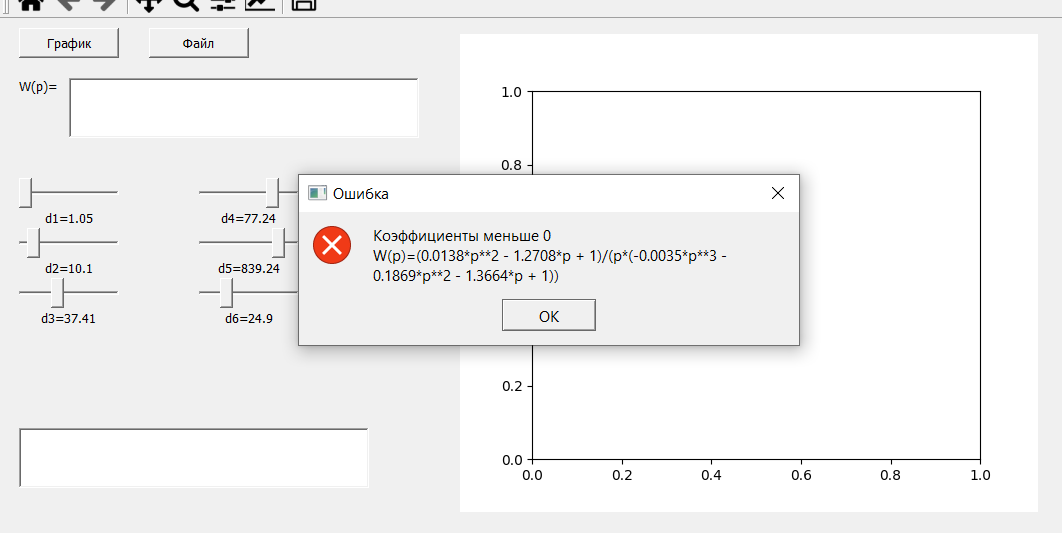


Рисунок 3 – Примеры работы программы

# Задание №4

## Постановка задачи

На систему подается произвольный сигнал, даны точки выходного сигнала. Необходимо произвести идентификацию системы, т. е. получить передаточную функцию, которая бы правильно реагировала на подаваемый сигнал, выдавая необходимый на выходе. Необходимо реализовать автоматический и ручной подбор узлов, вывод значения двух метрик (максимальное отклонение и дисперсия). Структуры для идентификации: 0/2, 2/2, 1/1, 3/3.

## Решение задачи

Для того чтобы решить эту задачу так же воспользуемся вещественным интерполяционным методом, но с тем условием, что входной сигнал заранее неизвестен.

Тогда функция может быть получена другим путем – на основе определения передаточной функции как отношения изображения выходного сигнала к изображению входного сигнала (при нулевых начальных условиях):

|  |  |  |
| --- | --- | --- |
|  |  | (16) |

При этом соотношение «вход-выход» системы принимает вид .

Для того чтобы по полученной передаточной функции получить выходной сигнал, для того чтобы сравнить его с заданным необходимо так же получить изображение входного сигнала . После этого перемножением передаточной функции на изображение входного сигнала можно получить изображение выходного сигнала.

В начале работы программы из файла считываются точки входного сигнала, выводятся на график и находится изображение . Далее также из файла считываются точки выходного сигнала, выводятся на график и по выбранной структуре находится передаточная функция . После этого находится изображение выходного сигнала , путем перемножения передаточной функции и входного сигнала. Для него применятся численное обращение преобразования Лапласа и основная теорема вычетов, и полученное выражение y(t) выводится на график.

## Алгоритмы

**korni3–** Функция для вычисления корней полинома 3 степени

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Вид |
| K | Массив коэффициентов полинома | float | Входной |
| a,b,c | 1,2,3 коэффициент соответственно. Деленные на 0 коэффициент | float | Промежуточный |
| Q,R | Выражения составленные из a,b,c | float | Промежуточный |
| S | Дискриминант 3 степени | float | Промежуточный |
| fi | Угол для исчисления корней | float | Промежуточный |
| x1,x2,x3 | Корни полинома | float/  complex | Выходной |



конец

начало

K=[1,K[1]/K[0],K[2]/K[0],K[3]/K[0]]

a=K[1];b=K[2];c=K[3];

Q=(a\*\*2-3\*b)/9

R=(2\*a\*\*3-9\*a\*b+27\*c)/54

S=(Q\*\*3-R\*\*2)



S>0

fi=(arccos(R/Q\*\*(1.5)))/3

x1=-2\*(Q)\*\*(0.5)\* cos(fi) - a/3

x2 = -2 \* (Q) \*\* (0.5) \*cos(fi+2\* pi/3) - a / 3

x3 = -2 \* (Q) \*\* (0.5) \*cos(fi - 2 \*pi / 3) - a/ 3

Q>0



fi=(arccosh(|R|)/|Q|\*\*(1.5)))/3

x1= -2\* sign(R) \*|Q|\*\*(0.5) \*cosh(fi) - a / 3

x2 = sign(R) \*|Q|\*\* (0.5)\*cosh(fi)- a /3 +i\*(3\*|Q|\*\* (0.5)\* sinh(fi))

x3 = sign(R) \*|Q|\*\* (0.5)\*cosh(fi)- a /3 -i\*(3\*|Q|\*\* (0.5)\* sinh(fi))

Q<0

fi = (arcsinh(|R| / |Q| \*\* (1.5))) / 3  
x1 = -2 \* sign(R) \* |Q| \*\* (0.5) \* sinh(fi) - a / 3  
x2 = sign(R) \* |Q| \*\* (0.5) \* sinh(fi) - a / 3+i\*(3 \* |Q|) \*\* (0.5) \*cosh(fi))  
x3 = sign(R) \* |Q| \*\* (0.5) \* sinh(fi) - a / 3-i\*(3 \* |Q|) \*\* (0.5) \*cosh(fi))

x1=-(c-a\*\*3/27)\*\*(1/3)-a/3  
x2=-(a+x1)\*0.5+i\*0.5\*|(a-3\*x1)\*(a+x1)-4\*b)\*\*(0.5))|  
x3=-(a+x1)\*0.5-i\*0.5\*|(a-3\*x1)\*(a+x1)-4\*b)\*\*(0.5))|



S=0

x1 = -2 \* (R) \*\* (1/3) - a / 3  
x2 = (R) \*\* (1 / 3) - a / 3  
x3 = (R) \*\* (1 / 3) - a / 3

**Korni2–** Функция для вычисления корней полинома 2 степени для структуры 0/2

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Вид |
| K | Массив коэффициентов полинома | float | Входной |
| a,b | 1,2 коэффициент соответственно.  Из структуры K[0]/( K[1]\*d^2+ K[2]\*d+1) | float | Промежуточный |
| dis | Дискриминант | float | Промежуточный |
| x1,x2 | Корни полинома | float/  complex | Выходной |



конец

начало

a = K[1]

b = K[2]

dis = b \*\* 2 - 4 \* a



dis>=0

x1 = (-b + |dis|\*\* (0.5)) / (2 \* a)  
x2 = (-b - |dis|\*\* (0.5)) / (2 \* a)

x1 = -b / (2 \* a)+i\*|dis|\*\* (0.5) / (2 \* a))  
x2 = -b / (2 \* a)-i\*|dis|\*\* (0.5) / (2 \* a))

**PF–** Функция для вычисления передаточной по точкам, адаптировано если t переменное

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Вид |
| A | Матрица точек t,x или t,x,y | float | Входной |
| d | Дельта | float | Входной |
| X | X(p), если y(t) не задано, то W(p) | float | Промежуточный |
| Y | Y(p) | float | Промежуточный |
| t | Точки по времени | float | Промежуточный |
| dt | Шаг по времени | float | Промежуточный |
| W | Передаточная | float | Выходной |



конец

начало

X=0;Y = 0; t=A[:, 0]



I==0

dt=t[i+1]-t[i]

dt=t[i]-t[i-1]

i от 0 до длины t

X += A[i, 1] \* exp(-d \* t[i]) \* dt

Y += A[i, 2] \*exp(-d \* t[i]) \* dt

Длина(A[0,:])=3



W = Y / X

Длина(A[0,:])=3

W = X

**rety–** Функция для вычисления выходного сигнала методом вычетов

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Вид |
| K | Массив коэффициентов Y(p) | float | Входной |
| Kx | Массив коэффициентов X(p) | float | Входной |
| n | структура | string | Входной |
| t | Точки по времени | float | входной |
| K1 | Коэффициенты для вычисления корней или для m | float | Промежуточный |
| y1,y2,y3 | Корни знаменателя Y(p) | float | Промежуточный |
| x1,x2 | Корни знаменателя X(p) | float | Промежуточный |
| A1-A5 | Коэффициенты в конечной формуле метода вычетов | float | Промежуточный |
| m | переменная для вычисления A1-A5 | float | Промежуточный |
| yy | сигнал Y(t) | float | Выходной |



начало

x1,x2=korni2(Kx)

n=**'3/3'**



K1=[K[4],K[5],K[6],1]

y1,y2,y3=korni3(K1)



n=**'1/1'**



y1 = -1/K[2]  
K1=[K[2]]



n=**'2/2'**



K1 = [K[3]]  
y1, y2 = korni2([K[2],K[3],K[4]])



n=**'0/2'**



y1, y2 = korni2(K)  
K1=[K[1]]

m=Kx[0]/(K1[0]\*Kx[1])



n=**'3/3'**

A1= (K[0]\*y1\*\*3+K[1]\*y1\*\*2+K[2]\*y1+K[3])\*m/ ((y1- y2)\*(y1- y3)\*(y1-x1)\*(y1-x2))  
A2 = (K[0]\*y2\*\*3+K[1]\*y2\*\*2+K[2]\*y2+K[3])\*m/ ((y2- y1)\*(y2-y3)\*(y2-x1)\*(y2-x2))  
A3 = (K[0]\*y3\*\*3+K[1]\*y3\*\*2+K[2]\*y3+K[3])\*m/ ((y3- y2)\*(y3-y1)\*(y3-x1)\*(y3-x2))  
A4 = (K[0]\*x1\*\*3+K[1]\*x1\*\*2+K[2]\*x1+K[3])\*m/ ((x1-y2) \* (x1-y3)\*(x1-y1)\*(x1-x2))  
A5 = (K[0]\*x2\*\*3+K[1]\*x2\*\*2+K[2]\*x2+K[3])\*m/ ((x2-y2)\*(x2-y3)\*(x2-x1)\*(x2 - y1))

yy(t)=A1\*exp(y1\*t)+A2\*exp(y2\*t) +A3\*exp(y3\*t) +A4\*exp(x1\*t) +A5\*exp(x2\*t)



n=**'2/2'**

A1 = (K[0] \* y1 \*\* 2 + K[1] \* y1 + K[2]) \* m / ((y1 - y2) \* (y1 - x1) \* (y1 - x2))  
A2 = (K[0] \* y2 \*\* 2 + K[1] \* y2 + K[2]) \* m / ((y2 - y1) \* (y2 - x1) \* (y2 - x2))  
A3 = (K[0] \* x1 \*\* 2 + K[1] \* x1 + K[2]) \* m / ((x1 - y2) \* (x1 - y1) \* (x1 - x2))  
A4 = (K[0] \* x2 \*\* 2 + K[1] \* x2 + K[2]) \* m / ((x2 - y2) \* (x2 - x1) \* (x2 - y1))  
yy(t)=A1\*exp(y1\*t)+A2\*exp(y2\*t) +A3\*exp(x1\*t) +A4\*exp(x2\*t)

A1 = (K[0] \* y1 + K[1]) \* m / ((y1 - x1) \* (y1 - x2))  
A2 = (K[0] \* x1 + K[1]) \* m / ((x1 - y1) \* (x1 - x2))  
A3 = (K[0] \* x2 + K[1]) \* m / ((x2 - x1) \* (x2 - y1))  
yy(t)=A1\*exp(y1\*t)+A2\*exp(x1\*t) +A3\*exp(x2\*t)



n=**'1/1'**

A1 = (K[0]) \* m / ((y1 - y2) \* (y1 - x1) \* (y1 - x2))  
A2 = (K[0]) \* m / ((y2 - y1) \* (y2 - x1) \* (y2 - x2))  
A3 = (K[0]) \* m / ((x1 - y2) \* (x1 - y1) \* (x1 - x2))  
A4 = (K[0]) \* m / ((x2 - y2) \* (x2 - x1) \* (x2 - y1))  
yy(t)=A1\*exp(y1\*t)+A2\*exp(y2\*t) +A3\*exp(x1\*t) +A4\*exp(x2\*t)



n=**'0/2'**

конец

retx(t,K) – вычисляется вход при структуре 0/2

automatx(A) – автоматически подбираются узлы (дельты) для расчета передаточной и входа

automaty(A,n,Kx) – автоматически подбираются узлы (дельты) для расчета передаточной и выхода

OBG(d, A, n,Kx) – в ручную вычисляет выход Y(t)

Ksolve(d, t,x, n,hyst=0) – вычисляет коэффициенты передаточной, решая СЛАУ при определенной структуре

## Код

test.py

**import** time  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** scipy **import** signal  
**import** math  
**from** sympy **import** \*  
**import** mpmath **as** mp  
**def** korni3(K):  
 K=[1,K[1]/K[0],K[2]/K[0],K[3]/K[0]]  
 a=K[1]  
 b=K[2]  
 c=K[3]  
 Q=(a\*\*2-3\*b)/9  
 R=(2\*a\*\*3-9\*a\*b+27\*c)/54  
 S=(Q\*\*3-R\*\*2).real  
 Q=Q.real  
 **if** S>0:  
 fi=(np.arccos(R/Q\*\*(1.5)))/3  
 x1=-2\*(Q)\*\*(0.5)\*np.cos(fi) - a/3  
 x2 = -2 \* (Q) \*\* (0.5) \* np.cos(fi+2\*np.pi/3) - a / 3  
 x3 = -2 \* (Q) \*\* (0.5) \* np.cos(fi - 2 \* np.pi / 3) - a / 3  
 **if** S < 0:  
 **if** Q>0:  
 fi=(np.arccosh(abs(R)/abs(Q)\*\*(1.5)))/3  
 x1= -2\*np.sign(R) \* abs(Q)\*\*(0.5) \* np.cosh(fi) - a / 3  
 x2 =complex(np.sign(R) \* abs(Q) \*\* (0.5)\* np.cosh(fi)- a / 3,(3\*abs(Q))\*\*(0.5)\*np.sinh(fi))  
 x3 =complex(np.sign(R) \* abs(Q) \*\* (0.5)\* np.cosh(fi)- a / 3,-(3\*abs(Q))\*\*(0.5)\*np.sinh(fi))  
 **if** Q < 0:  
 fi = (np.arcsinh(abs(R) / abs(Q) \*\* (1.5))) / 3  
 x1 = -2 \* np.sign(R) \* abs(Q) \*\* (0.5) \* np.sinh(fi) - a / 3  
 x2 = complex(np.sign(R) \* abs(Q) \*\* (0.5) \* np.sinh(fi) - a / 3, (3 \* abs(Q)) \*\* (0.5) \* np.cosh(fi))  
 x3 = complex(np.sign(R) \* abs(Q) \*\* (0.5) \* np.sinh(fi) - a / 3, -(3 \* abs(Q)) \*\* (0.5) \* np.cosh(fi))  
 **if** Q==0:  
 x1=-(c-a\*\*3/27)\*\*(1/3)-a/3  
 x2=complex(-(a+x1)\*0.5,0.5\*(abs((a-3\*x1)\*(a+x1)-4\*b)\*\*(0.5)))  
 x3 = complex(-(a + x1) \* 0.5, -0.5 \* (abs((a - 3 \* x1) \* (a + x1) - 4 \* b) \*\* (0.5)))  
 **if** S==0:  
 x1 = -2 \* (R) \*\* (1/3) - a / 3  
 x2 = (R) \*\* (1 / 3) - a / 3  
 x3 = (R) \*\* (1 / 3) - a / 3  
 **return** (x1,x2,x3)  
**def** korni2(K):  
 a = K[1]  
 b = K[2] *# b0/(a1d^2+a2d+1)* dis = b \*\* 2 - 4 \* a  
 **if** dis < 0:  
 x1 = complex(-b / (2 \* a), abs(dis) \*\* (0.5) / (2 \* a))  
 x2 = complex(-b / (2 \* a), -abs(dis) \*\* (0.5) / (2 \* a))  
 **else**:  
 x1 = (-b + abs(dis) \*\* (0.5)) / (2 \* a)  
 x2 = (-b - abs(dis) \*\* (0.5)) / (2 \* a)  
 **return** (x1,x2)  
**def** PF(A, d)X=0  
 Y = 0  
 t=A[:, 0]  
 **for** i **in** range(len(t)):  
 **if** (i==0): dt=t[i+1]-t[i]  
 **else**:dt=t[i]-t[i-1]  
 X += A[i, 1] \* np.exp(-d \* t[i]) \* dt  
 **if** len(A[0, :]) == 3:  
 Y += A[i, 2] \* np.exp(-d \* t[i]) \* dt  
 **if** len(A[0, :]) == 3:  
 W = Y / X  
 **else**:  
 W = X  
 **return** (W)  
**def** rety(t,K,Kx,n):  
 x1,x2=korni2(Kx)  
 **if** n==**'3/3'**:  
 K1=[K[4],K[5],K[6],1]  
 y1,y2,y3=korni3(K1)  
 **if** n==**'1/1'**:*#a0\*p+a1/a2\*p+1\*a0x/a1xp^2+a2x\*p+1* y1 = -1/K[2]  
 print(K[2],y1)  
 K1=[K[2]]  
 **if** n==**'2/2'**:  
 K1 = [K[3]]  
 y1, y2 = korni2([K[2],K[3],K[4]])  
 **if** n==**'0/2'**:  
 y1, y2 = korni2(K)  
 K1=[K[1]]  
 m=Kx[0]/(K1[0]\*Kx[1])  
 yy = []  
 **if** n == **'3/3'**:*#a0/a1p^2+a2\*p+1 \* a0x/a1xp^2+a2x\*p+1* A1 = (K[0]\*y1\*\*3+K[1]\*y1\*\*2+K[2]\*y1+K[3])\*m/ ((y1 - y2)\*(y1 - y3)\*(y1-x1)\*(y1-x2))  
 A2 = (K[0]\*y2\*\*3+K[1]\*y2\*\*2+K[2]\*y2+K[3])\*m/ ((y2 - y1)\*(y2 - y3)\*(y2-x1)\*(y2-x2))  
 A3 = (K[0]\*y3\*\*3+K[1]\*y3\*\*2+K[2]\*y3+K[3])\*m/ ((y3 - y2)\*(y3 - y1)\*(y3-x1)\*(y3-x2))  
 A4 = (K[0]\*x1\*\*3+K[1]\*x1\*\*2+K[2]\*x1+K[3]) \* m/ ((x1 - y2) \* (x1 - y3)\*(x1 - y1)\*(x1 - x2))  
 A5 = (K[0]\*x2\*\*3+K[1]\*x2\*\*2+K[2]\*x2+K[3]) \* m/ ((x2 - y2) \* (x2 - y3)\*(x2 - x1)\*(x2 - y1))  
 **for** i **in** t:  
 yy+=[(A1 \* np.exp(y1 \* i)+A2 \* np.exp(y2 \* i)+A3 \* np.exp(y3 \* i)+A4 \* np.exp(x1 \* i)+A5 \* np.exp(x2 \* i)).real]  
 **if** n == **'1/1'**:  
 A1 = (K[0] \* y1 + K[1]) \* m / ((y1 - x1) \* (y1 - x2))  
 A2 = (K[0] \* x1 + K[1]) \* m / ((x1 - y1) \* (x1 - x2))  
 A3 = (K[0] \* x2 + K[1]) \* m / ((x2 - x1) \* (x2 - y1))  
 yy = []  
 **for** i **in** t:  
 yy += [(A1 \* np.exp(y1 \* i) + A2 \* np.exp(x1 \* i) + A3 \* np.exp(x2 \* i)).real]  
 **if** n == **'0/2'**:  
 A1 = (K[0]) \* m / ((y1 - y2) \* (y1 - x1) \* (y1 - x2))  
 A2 = (K[0]) \* m / ((y2 - y1) \* (y2 - x1) \* (y2 - x2))  
 A3 = (K[0]) \* m / ((x1 - y2) \* (x1 - y1) \* (x1 - x2))  
 A4 = (K[0]) \* m / ((x2 - y2) \* (x2 - x1) \* (x2 - y1))  
 yy = []  
 **for** i **in** t:  
 yy += [(A1 \* np.exp(y1 \* i) + A2 \* np.exp(y2 \* i) + A3 \* np.exp(x1 \* i) + A4 \* np.exp(  
 x2 \* i)).real]  
 **if** n == **'2/2'**:  
 A1 = (K[0] \* y1 \*\* 2 + K[1] \* y1 + K[2]) \* m / ((y1 - y2) \* (y1 - x1) \* (y1 - x2))  
 A2 = (K[0] \* y2 \*\* 2 + K[1] \* y2 + K[2]) \* m / ((y2 - y1) \* (y2 - x1) \* (y2 - x2))  
 A3 = (K[0] \* x1 \*\* 2 + K[1] \* x1 + K[2]) \* m / ((x1 - y2) \* (x1 - y1) \* (x1 - x2))  
 A4 = (K[0] \* x2 \*\* 2 + K[1] \* x2 + K[2]) \* m / ((x2 - y2) \* (x2 - x1) \* (x2 - y1))  
 **for** i **in** t:  
 yy += [(A1 \* np.exp(y1 \* i) + A2 \* np.exp(y2 \* i) + A3 \* np.exp(x1 \* i) + A4 \* np.exp(  
 x2 \* i)).real]  
 **return** yy  
**def** retx(t,K):  
 b0=K[0]/K[1]  
 x1,x2=korni2(K)  
 A1 = b0 / (x1 - x2)  
 xx = []  
 **for** i **in** t:  
 xx += [(A1 \* (np.exp(x1 \* i) - np.exp(x2 \* i))).real]  
 **return** xx  
**def** automatx(A):  
 d = [0, 0, 0]  
 A=A[:,:2]  
 t=A[:,0]  
 x = A[:, 1]  
 ns = len(t)  
 minmax=100000  
 disp\_disp=100000  
 **for** d[0] **in** range(0, 100, 1):  
 d[0] = d[0] / 10  
 **if** minmax<0.1:  
 **break  
 for** d[1] **in** range(math.ceil(1 + d[0] \* 10), 100, 1):  
 d[1] = d[1] / 10  
 **for** d[2] **in** range(math.ceil(1 + d[1] \* 10), 100, 1):  
 d[2] = d[2] / 10  
 d=[d[0],d[1],d[2]]  
 K=Ksolve(d,A,**'0/2'**)  
 **if not** np.all(abs(K) == K):  
 **continue** xx=retx(t,K)  
 disp=0  
 **for** i **in** range(ns):  
 disp += (x[i] - xx[i]) \*\* 2  
 disp = (disp / (ns - 1)) \*\* (1 / 2)  
 maxn = max(abs(x - xx))  
 **if** maxn<=minmax **and** disp<=disp\_disp:  
 minmax=maxn  
 disp\_disp=disp  
 delta=[d[0],d[1],d[2]]  
 print(delta, minmax)  
 Kx = Ksolve(delta,A,**'0/2'**)  
 xx=retx(t,Kx)  
 Kx = np.around(Kx, decimals=4, out=**None**)  
 fp=fp\_K(Kx, **'0/2'**)  
 **return** xx,fp,Kx  
**def** automaty(A,n,Kx):  
 t = A[:, 0]  
 y = A[:, 2]  
 **if** n == **'3/3'**:  
 d = [0, 0, 0, 0, 0,0,0]  
 **if** n == **'2/2'**:  
 d = [0, 0, 0, 0, 0]  
 **if** n == **'0/2' or** n == **'1/1'**:  
 d = [0, 0, 0]  
 ns = len(t)  
 minmax=1000000  
 disp\_disp=1000000  
 delta=[]  
 **if** n == **'3/3'**:  
 **for** d[0] **in** range(0, 1, 1):  
 d[0] = d[0]/100  
 **for** d[1] **in** range(math.ceil(1 + d[0]\*100),2, 1):  
 d[1] = d[1]/100  
 **for** d[2] **in** range(math.ceil(1 + d[1]\*100), 5, 1):  
 d[2] = d[2]/100  
 **for** d[3] **in** range(math.ceil(1 + d[2]\*100), 20, 1):  
 d[3] = d[3]/100  
 **for** d[4] **in** range(math.ceil(1 + d[3]\*100), 30, 1):  
 d[4] = d[4]/100  
 **for** d[5] **in** range(math.ceil(1 + d[4]\*100), 30, 1):  
 d[5] = d[5]/100  
 **for** d[6] **in** range(math.ceil(1 + d[5]\*100), 30, 1):  
 d[6] = d[6]/100  
 d = [d[0], d[1], d[2], d[3], d[4],d[5],d[6]]  
 K = Ksolve(d, A, n)  
 **if not** np.all(abs(K) == K):  
 **continue** yy = rety(t, K, Kx, n)  
 disp = 0  
 **for** i **in** range(ns):  
 disp += (y[i] - yy[i]) \*\* 2  
 disp = (disp / (ns - 1)) \*\* (1 / 2)  
 maxn = max(abs(y - yy))  
 **if** maxn <= minmax **and** disp <= disp\_disp:  
 minmax = maxn  
 disp\_disp = disp  
 delta = [d[0], d[1], d[2], d[3], d[4],d[5],d[6]]  
 print(delta, minmax, disp\_disp)  
 **if** n == **'2/2'**:  
 T=10  
 **for** d[0] **in** range(0, 1, 1):  
 d[0] = d[0]/T  
 **for** d[1] **in** range(math.ceil(1 + d[0]\*T), 15, 1):  
 d[1] = d[1]/T  
 **for** d[2] **in** range(math.ceil(1 + d[1]\*T), 20, 1):  
 d[2] = d[2]/T  
 **for** d[3] **in** range(math.ceil(1+ d[2]\*T), 25, 1):  
 d[3] = d[3]/T  
 **for** d[4] **in** range(math.ceil(1 + d[3]\*T), 30, 1):  
 d[4] = d[4]/T  
 d = [d[0], d[1], d[2],d[3], d[4]]  
 K = Ksolve(d, A, n)  
 **if not** np.all(abs(K) == K):  
 **continue** yy = rety(t, K, Kx, n)  
 disp = 0  
 **for** i **in** range(ns):  
 disp += (y[i] - yy[i]) \*\* 2  
 disp = (disp / (ns - 1)) \*\* (1 / 2)  
 maxn = max(abs(y - yy))  
 **if** maxn <= minmax **and** disp <= disp\_disp:  
 minmax = maxn  
 disp\_disp = disp  
 delta = [d[0], d[1], d[2],d[3], d[4]]  
 print(delta, minmax, disp\_disp)  
 *#(delta, maxn)* **elif** n == **'0/2' or** n == **'1/1'**:  
 **for** d[0] **in** range(1, 2, 1):  
 d[0] = d[0]/10  
 **for** d[1] **in** range(math.ceil(1 + d[0]\*10),35, 1):  
 d[1] = d[1]/10  
 **for** d[2] **in** range(math.ceil(1 + d[1]\*10),65, 1):  
 d[2] = d[2]/10  
 d=[d[0],d[1],d[2]]  
 K = Ksolve(d, A, n)  
 **if not** np.all(abs(K) == K):  
 **continue** yy = rety(t, K, Kx, n)  
 disp = 0  
 **for** i **in** range(ns):  
 disp += (y[i] - yy[i]) \*\* 2  
 disp = (disp / (ns - 1)) \*\* (1 / 2)  
 maxn = max(abs(y - yy))  
 **if** maxn <= minmax **and** disp <= disp\_disp:  
 minmax = maxn  
 disp\_disp = disp  
 delta = [d[0],d[1],d[2]]  
 print(delta, minmax,disp\_disp)  
 **if** delta==[]:  
 **return** 0,0,0,0  
 K = Ksolve(delta, A, n)  
 yy = rety(t, K, Kx, n)  
 K = np.around(K, decimals=4, out=**None**)  
 minmax = np.around(minmax, decimals=4, out=**None**)  
 disp\_disp = np.around(disp\_disp, decimals=4, out=**None**)  
 fp=fp\_K(K, n)  
 **return** yy,minmax,disp\_disp,fp,d  
**def** Ksolve(d, A, n): *# вычисление коэффициентов* Matr = []  
 b = []  
 **for** i **in** range(len(d)):  
 W = PF(A, d[i])  
 **if** n == **'3/3'**:  
 Matr += [[d[i] \*\* 3, d[i] \*\* 2, d[i], 1, -W \* d[i] \*\* 3, -W \* d[i] \*\* 2,-W \* d[i]]]  
 **if** n == **'2/2'**:  
 Matr += [[d[i] \*\* 2, d[i], 1, -W \* d[i] \*\* 2, -W \* d[i]]]  
 **if** n == **'0/2'**:  
 Matr += [[1, -W \* d[i] \*\* 2, -W \* d[i]]]  
 **if** n == **'1/1'**:  
 Matr += [[d[i], 1, -W \* d[i]]] *#W=a0\*p+a1/(a2\*p+1)* b += [W]  
 K = np.linalg.solve(Matr, b)  
 **return** K  
**def** fp\_K(K,n):  
 **if** n == **'3/3'**:  
 fp = **lambda** p: (K[0] \* p \*\* 3 + K[1] \* p \*\* 2 + K[2] \* p + K[3]) / (  
 K[4] \* p \*\* 3 + K[5] \* p \*\* 2 + K[6] \* p + 1)  
 **if** n == **'2/2'**:  
 fp = **lambda** p: (K[0] \* p \*\* 2 + K[1] \* p + K[2]) / (K[3] \* p \*\* 2 + K[4] \* p + 1)  
 **if** n == **'0/2'**:  
 fp = **lambda** p: (K[0]) / (K[1] \* p \*\* 2 + K[2] \* p + 1)  
 **if** n == **'1/1'**:  
 fp = **lambda** p: (K[0] \* p + K[1]) / (K[0] \* p + 1)  
 **return** fp  
**def** OBG(d, A, n,Kx): *# поиск переходной характеристики и получение передаточной* t = A[:, 0]  
 x = A[:, 1]  
 y = A[:, 2]  
 ns=len(y)  
 K=Ksolve(d, A, n)  
 K = np.around(K, decimals=4, out=**None**)  
 p = symbols(**'p'**, positive=true)  
 fp=fp\_K(K,n)  
 **if not** np.all(abs(K) == K):  
 **return** 0, 0, fp  
 yy = rety(t, K, Kx, n)  
 sigma = 0  
 **for** i **in** range(ns):  
 sigma += (y[i] - yy[i]) \*\* 2  
 sigma = (sigma / (ns - 1)) \*\* (1 / 2)  
 deltahmax = max(abs(y - yy))  
 sigma = np.around(sigma, decimals=4, out=**None**)  
 deltahmax = np.around(deltahmax, decimals=4, out=**None**)  
 otv = **'Дисперсия = '** + str(sigma) + **'\ndeltahmax = '** + str(deltahmax)  
 **return** (otv, yy, fp)

Main:

**import** sys  
**from** PyQt5 **import** QtCore, QtWidgets  
**from** PyQt5.QtCore **import** Qt  
**from** PyQt5.QtWidgets **import** \*  
**from** matplotlib.backends.backend\_qt5agg **import** FigureCanvasQTAgg, NavigationToolbar2QT **as** NavigationToolbar  
**from** matplotlib.figure **import** Figure  
**import** pandas **as** pd  
**import** numpy **as** np  
  
**from** test **import** \*  
**from** mplwidget **import** MplWidget  
  
**class** MplCanvas(FigureCanvasQTAgg):  
  
 **def** \_\_init\_\_(self, parent=**None**, width=5, height=4, dpi=100):  
 fig = Figure(figsize=(width, height), dpi=dpi)  
 self.axes = fig.add\_subplot(111)  
 super(MplCanvas, self).\_\_init\_\_(fig)  
  
  
**class** MainWindow(QtWidgets.QMainWindow):  
  
 **def** \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(MainWindow, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.initUI()  
  
 **def** initUI(self):  
 self.setObjectName(**"MainWindow"**)  
 self.resize(1100, 700)  
 self.centralwidget = QtWidgets.QWidget(self)  
 self.centralwidget.setObjectName(**"centralwidget"**)  
 self.MplWidget = MplWidget(self.centralwidget)  
 self.MplWidget.setGeometry(QtCore.QRect(450, 5, 600, 500))  
 self.MplWidget.setMinimumSize(QtCore.QSize(450, 350))  
 self.MplWidget.setObjectName(**"MplWidget"**)  
 self.lblc = QLabel(self)  
 self.lblc.move(20, 400)  
 self.lblc.setText(**'Структура:'**)  
 self.combo = QComboBox(self)  
 self.combo.addItem(**'3/3'**)  
 self.combo.addItem(**'2/2'**)  
 self.combo.addItem(**'1/1'**)  
 self.combo.addItem(**'0/2'**)  
 self.combo1 = QComboBox(self)  
 self.combo1.addItem(**'Ползунком'**)  
 self.combo1.addItem(**'В поле'**)  
 self.combo1.move(350, 200)  
 self.combo.move(100, 400)  
 self.setCentralWidget(self.centralwidget)  
 self.addToolBar(NavigationToolbar(self.MplWidget.canvas, self))  
 self.rez = QTextEdit(self)  
 self.rez.move(70, 100)  
 self.rez.resize(350, 60)  
 self.rez2 = QTextEdit(self)  
 self.rez2.move(70, 600)  
 self.rez2.resize(350, 60)  
 self.lbl = QLabel(self)  
 self.lbl.move(20, 100)  
 self.lbl.setText(**'W(p)='**)  
 self.lbl.adjustSize()  
 self.lbl2 = QLabel(self)  
 self.lbl2.move(20, 600)  
 self.lbl2.setText(**'x(p)='**)  
 self.lbl2.adjustSize()  
 self.lotv = QTextEdit(self)  
 self.lotv.move(20, 450)  
 self.lotv.resize(350, 120)  
 **global** filename  
 filename = **'file.csv'** *# добавлене кнопок* btn1 = QPushButton(**"График"**, self)  
 btn1.move(20, 50)  
 btn2 = QPushButton(**"Файл"**, self)  
 btn2.move(150, 50)  
 *# если нажали на кнопку выполнить функции* btn1.clicked.connect(self.update\_graph2)  
 btn2.clicked.connect(self.getFileName)  
 self.statusBar()  
 self.setWindowTitle(**'ХвещукОА'**)  
 btn3 = QPushButton(**"Применить структуру"**, self)  
 btn3.setGeometry(250, 400, 150, 35)  
 btn3.clicked.connect(self.updatesld)  
 btn4 = QPushButton(**"Автоматический ввод"**, self)  
 btn4.setGeometry(450, 600, 150, 35)  
 btn4.clicked.connect(self.update)  
 *# если нажали на кнопку выполнить функции* t = 1000  
 t1 = 5  
 r = 0  
 **global** p  
 p = symbols(**'p'**)  
 *#1. Можно ли создать какой нибудь массив из sld чтобы не копипастить код постоянно?* self.sld1 = QSlider(Qt.Horizontal, self)  
 self.sld1.setRange(r, t)  
 self.sld1.setFocusPolicy(Qt.NoFocus)  
 self.sld1.setPageStep(t1)  
 self.sld1.move(20, 200)  
 self.sld1.valueChanged.connect(self.updateLabel1)  
 self.ld1 = QLabel(**'d1='** + str(r / 100), self)  
 self.ld1.setAlignment(Qt.AlignCenter | Qt.AlignVCenter)  
 self.ld1.setMinimumWidth(30)  
 self.ld1.move(20, 225)  
 self.sld2 = QSlider(Qt.Horizontal, self)  
 self.sld2.setRange(r, t)  
 self.sld2.setFocusPolicy(Qt.NoFocus)  
 self.sld2.setPageStep(t1)  
 self.sld2.move(20, 250)  
 self.sld2.valueChanged.connect(self.updateLabel2)  
 self.ld2 = QLabel(**'d2='** + str(r / 100), self)  
 self.ld2.setAlignment(Qt.AlignCenter | Qt.AlignVCenter)  
 self.ld2.setMinimumWidth(30)  
 self.ld2.move(20, 275)  
 self.sld3 = QSlider(Qt.Horizontal, self)  
 self.ld3 = QLabel(**'d3='** + str(r / 100), self)  
 self.ld4 = QLabel(**'d4='** + str(r / 100), self)  
 self.ld5 = QLabel(**'d5='** + str(r / 100), self)  
 self.ld6 = QLabel(**'d6='** + str(r / 100), self)  
 self.ld7 = QLabel(**'d7='** + str(r / 100), self)  
 self.sld4 = QSlider(Qt.Horizontal, self)  
 self.sld5 = QSlider(Qt.Horizontal, self)  
 self.sld6 = QSlider(Qt.Horizontal, self)  
 self.sld7 = QSlider(Qt.Horizontal, self)  
 self.sld3.setRange(r, t)  
 self.sld3.setFocusPolicy(Qt.NoFocus)  
 self.sld3.setPageStep(t1)  
 self.sld3.move(20, 300)  
 self.ld3.setAlignment(Qt.AlignCenter | Qt.AlignVCenter)  
 self.ld3.setMinimumWidth(30)  
 self.ld3.move(20, 325)  
 self.sld3.valueChanged.connect(self.updateLabel3)  
 self.sld4.setRange(r, t)  
 self.sld4.setFocusPolicy(Qt.NoFocus)  
 self.sld4.setPageStep(t1)  
 self.sld4.move(200, 200)  
 self.ld4.setAlignment(Qt.AlignCenter | Qt.AlignVCenter)  
 self.ld4.setMinimumWidth(30)  
 self.ld4.move(200, 225)  
 self.sld4.valueChanged.connect(self.updateLabel4)  
 self.sld5.setRange(r, t)  
 self.sld5.setFocusPolicy(Qt.NoFocus)  
 self.sld5.setPageStep(t1)  
 self.sld5.move(200, 250)  
 self.ld5.setAlignment(Qt.AlignCenter | Qt.AlignVCenter)  
 self.ld5.setMinimumWidth(30)  
 self.ld5.move(200, 275)  
 self.sld5.valueChanged.connect(self.updateLabel5)  
 self.sld6.setRange(r, t)  
 self.sld6.setFocusPolicy(Qt.NoFocus)  
 self.sld6.setPageStep(t1)  
 self.sld6.move(200, 300)  
 self.ld6.setAlignment(Qt.AlignCenter | Qt.AlignVCenter)  
 self.ld6.setMinimumWidth(30)  
 self.ld6.move(200, 325)  
 self.sld6.valueChanged.connect(self.updateLabel6)  
 self.sld7.setRange(r, t)  
 self.sld7.setFocusPolicy(Qt.NoFocus)  
 self.sld7.setPageStep(t1)  
 self.sld7.valueChanged.connect(self.updateLabel7)  
 self.sld7.move(20, 350)  
 self.ld7.setAlignment(Qt.AlignCenter | Qt.AlignVCenter)  
 self.ld7.setMinimumWidth(30)  
 self.ld7.move(20, 375)  
 self.d1 = QLineEdit(self)  
 self.d1.move(50, 200)  
 self.d1.resize(70, 30)  
 self.d2 = QLineEdit(self)  
 self.d2.move(50, 250)  
 self.d2.resize(70, 30)  
 self.d3 = QLineEdit(self)  
 self.d3.move(50, 300)  
 self.d3.resize(70, 30)  
 self.d4 = QLineEdit(self)  
 self.d4.move(180, 200)  
 self.d4.resize(70, 30)  
 self.d5 = QLineEdit(self)  
 self.d5.move(180, 250)  
 self.d5.resize(70, 30)  
 self.d7 = QLineEdit(self)  
 self.d7.move(50, 350)  
 self.d7.resize(70, 30)  
 self.d6 = QLineEdit(self)  
 self.d6.move(180, 300)  
 self.d6.resize(70, 30)  
 self.ldd1 = QLabel(self)  
 self.ldd1.move(20, 200)  
 self.ldd1.setText(**'d1='**)  
 self.ldd1.adjustSize()  
 self.ldd2 = QLabel(self)  
 self.ldd2.move(20, 250)  
 self.ldd2.setText(**'d2='**)  
 self.ldd2.adjustSize()  
 self.ldd3 = QLabel(self)  
 self.ldd3.move(20, 300)  
 self.ldd3.setText(**'d3='**)  
 self.ldd3.adjustSize()  
 self.ldd4 = QLabel(self)  
 self.ldd4.move(150, 200)  
 self.ldd4.setText(**'d4='**)  
 self.ldd4.adjustSize()  
 self.ldd5 = QLabel(self)  
 self.ldd5.move(150, 250)  
 self.ldd5.setText(**'d5='**)  
 self.ldd5.adjustSize()  
 self.ldd6 = QLabel(self)  
 self.ldd6.move(150, 300)  
 self.ldd6.setText(**'d6='**)  
 self.ldd6.adjustSize()  
 self.ldd7 = QLabel(self)  
 self.ldd7.move(20, 350)  
 self.ldd7.setText(**'d7='**)  
 self.ldd7.adjustSize()  
 self.yjas()  
 self.statusBar()  
 self.setWindowTitle(**'ХвещукОА'**)  
 self.show()  
 *# если нажали на кнопку выполнить функции* **def** yjas(self):  
 self.d1.hide()  
 self.ldd1.hide()  
 self.d2.hide()  
 self.ldd2.hide()  
 self.d3.hide()  
 self.ldd3.hide()  
 self.d4.hide()  
 self.ldd4.hide()  
 self.d5.hide()  
 self.ldd5.hide()  
 self.d6.hide()  
 self.ldd6.hide()  
 self.d7.hide()  
 self.ldd7.hide()  
 self.sld1.hide()  
 self.ld1.hide()  
 self.sld2.hide()  
 self.ld2.hide()  
 self.sld3.hide()  
 self.ld3.hide()  
 self.sld4.hide()  
 self.ld4.hide()  
 self.sld5.hide()  
 self.ld5.hide()  
 self.sld6.hide()  
 self.ld6.hide()  
 self.sld7.hide()  
 self.ld7.hide()  
 **def** update(self):  
 self.lotv.clear()  
 self.MplWidget.canvas.axes.clear()  
 A = np.loadtxt(filename)  
 A[:,0] = A[:,0]\*4  
 A[:, 1] = np.sin(A[:, 0])  
 n = self.combo.currentText()  
 xx, xpf, Kx = automatx(A)  
 self.rez2.setText(str(xpf(p)))  
 self.MplWidget.canvas.axes.plot(A[:, 0], A[:, 1], **'go'**)  
 self.MplWidget.canvas.axes.plot(A[:, 0], xx, **'g'**)  
 self.MplWidget.canvas.draw()  
 yy, minmax, disp\_disp, fp,d = automaty(A, n, Kx)  
 otv= **'Автоматический ввод: '** +**'Дисперсия = '** + str(disp\_disp) + **'\ndeltahmax = '** + str(minmax)+**'d= '**+str(d)+**'W= '**+str(fp(p))  
 self.MplWidget.canvas.axes.plot(A[:, 0], A[:, 2], **'ro'**)  
 self.MplWidget.canvas.axes.plot(A[:, 0], yy, **'r'**)  
 self.MplWidget.canvas.axes.legend((**'x(t)'**, **'x(t)получ'**,**'y(t)'**, **'y(t)получ'**), loc=**'lower right'**)  
 self.MplWidget.canvas.axes.set\_xlabel(**'t'**)  
 self.MplWidget.canvas.axes.set\_ylabel(**'y(t)'**)  
 self.MplWidget.canvas.axes.grid()  
 self.MplWidget.canvas.draw()  
 self.lotv.setText(otv)  
 self.rez.setText(str(fp(p)))  
 **def** update\_graph2(self):  
 self.lotv.clear()  
 self.rez2.clear()  
 self.MplWidget.canvas.axes.clear()  
  
 A=np.loadtxt(filename)  
 A[:,0] = A[:,0]\*4  
 A[:, 1] = np.sin(A[:, 0])  
 d=self.dfrom  
 **if not** len(d) == len(set(d)):  
 QMessageBox.critical(self, **"Ошибка "**, **"Дельты не должны равняться друг другу"**, QMessageBox.Ok)  
 **return** n = self.combo.currentText()  
 xx, xpf, Kx=automatx(A)  
 self.rez2.setText(str(xpf(p)))  
 self.MplWidget.canvas.axes.plot(A[:, 0], A[:, 1], **'go'**)  
 self.MplWidget.canvas.axes.plot(A[:, 0], xx, **'g'**)  
 self.MplWidget.canvas.draw()  
 otv1, yy, fp = OBG(d, A, n,Kx)  
 self.rez.setText(str(fp(p)))  
 **if** yy == 0:  
 QMessageBox.critical(self, **"Ошибка "**, **"Коэффициенты меньше 0"**, QMessageBox.Ok)  
 self.lotv.clear()  
 self.rez.clear()  
 self.MplWidget.canvas.axes.clear()  
 self.MplWidget.canvas.draw()  
 **return** otv = **'Ручной ввод: '** + otv1  
 self.lotv.setText(otv)  
 self.MplWidget.canvas.axes.plot(A[:, 0], A[:, 2], **'ro'**)  
 self.MplWidget.canvas.axes.plot(A[:, 0], yy, **'r'**)  
 self.MplWidget.canvas.axes.legend((**'x(t)'**, **'x(t)получ'**,**'y(t)'**, **'y(t)получ'**), loc=**'lower right'**)  
 self.MplWidget.canvas.axes.set\_xlabel(**'t'**)  
 self.MplWidget.canvas.axes.set\_ylabel(**'y(t)'**)  
 self.MplWidget.canvas.axes.grid()  
 self.MplWidget.canvas.draw()  
 **def** getFileName(self):  
 **global** filename  
 filename, filetype = QFileDialog.getOpenFileName(self, **"All Files(\*.\*)"**)  
 *# print (os.path.basename(filename))* **def** updateLabel1(self, value): self.ld1.setText(**'d1='** + str(value / 100))  
 **def** updateLabel2(self, value): self.ld2.setText(**'d2='** + str(value / 100))  
 **def** updateLabel3(self, value): self.ld3.setText(**'d3='** + str(value / 100))  
 **def** updateLabel4(self, value): self.ld4.setText(**'d4='** + str(value / 100))  
 **def** updateLabel5(self, value): self.ld5.setText(**'d5='** + str(value / 100))  
 **def** updateLabel6(self, value): self.ld6.setText(**'d6='** + str(value / 100))  
 **def** updateLabel7(self, value): self.ld7.setText(**'d7='** + str(value / 100))  
 @property  
 **def** dfrom(self):  
 n = self.combo.currentText()  
 c = self.combo1.currentText()  
 **if** c==**'Ползунком'**:  
 d1 = self.ld1.text()  
 d1 = float(d1[3:])  
 d2 = self.ld2.text()  
 d2 = float(d2[3:])  
 d3 = self.ld3.text()  
 d3 = float(d3[3:])  
 d=[d1,d2,d3]  
 **if** n == **'3/3'**:  
 d4 = self.ld4.text()  
 d4 = float(d4[3:])  
 d5 = self.ld5.text()  
 d5 = float(d5[3:])  
 d6 = self.ld6.text()  
 d6 = float(d6[3:])  
 d7 = self.ld7.text()  
 d7 = float(d7[3:])  
 d+=[d4,d5,d6,d7]  
 **if** n == **'2/2'**:  
 d4 = self.ld4.text()  
 d4 = float(d4[3:])  
 d5 = self.ld5.text()  
 d5 = float(d5[3:])  
 d += [d4,d5]  
 **if** c == **'В поле'**:  
 d1 = float(self.d1.text())  
 d2 = float(self.d2.text())  
 d3 = float(self.d3.text())  
 d = [d1, d2,d3]  
 **if** n == **'3/3'**:  
 d4 = float(self.d4.text())  
 d5 = float(self.d5.text())  
 d6 = float(self.d6.text())  
 d7 = float(self.d7.text())  
 d += [d4, d5, d6,d7]  
 **if** n == **'2/2'**:  
 d4 = float(self.d4.text())  
 d5 = float(self.d5.text())  
 d += [d4,d5]  
 **return** d  
 **def** updatesld(self):  
 self.yjas()  
 n = self.combo.currentText()  
 c=self.combo1.currentText()  
 **if** c==**'Ползунком'**:  
 self.sld1.show()  
 self.ld1.show()  
 self.sld2.show()  
 self.ld2.show()  
 self.sld3.show()  
 self.ld3.show()  
 **if** n == **'3/3'**:  
 self.sld4.show()  
 self.ld4.show()  
 self.sld5.show()  
 self.ld5.show()  
 self.sld6.show()  
 self.ld6.show()  
 self.sld7.show()  
 self.ld7.show()  
 **if** n == **'2/2'**:  
 self.sld4.show()  
 self.ld4.show()  
 self.sld5.show()  
 self.ld5.show()  
 self.sld6.hide()  
 self.ld6.hide()  
 self.sld7.hide()  
 self.ld7.hide()  
 **if** n == **'0/2' or** n == **'1/1'**:  
 self.sld4.hide()  
 self.ld4.hide()  
 self.sld5.hide()  
 self.ld5.hide()  
 self.sld6.hide()  
 self.ld6.hide()  
 self.sld7.hide()  
 self.ld7.hide()  
 **if** c == **'В поле'**:  
 self.d1.show()  
 self.ldd1.show()  
 self.d2.show()  
 self.ldd2.show()  
 self.d3.show()  
 self.ldd3.show()  
 **if** n == **'3/3'**:  
 self.d4.show()  
 self.ldd4.show()  
 self.d5.show()  
 self.ldd5.show()  
 self.d6.show()  
 self.ldd6.show()  
 self.d7.show()  
 self.ldd7.show()  
 **if** n == **'2/2'**:  
 self.d3.show()  
 self.ldd3.show()  
 self.d4.show()  
 self.ldd4.show()  
 self.d5.show()  
 self.ldd5.show()  
 self.d6.hide()  
 self.ldd6.hide()  
 self.d7.hide()  
 self.ldd7.hide()  
 **if** n == **'0/2' or** n == **'1/1'**:  
 self.d4.hide()  
 self.ldd4.hide()  
 self.d5.hide()  
 self.ldd5.hide()  
 self.d6.hide()  
 self.ldd6.hide()  
 self.d7.hide()  
 self.ldd7.hide()  
*# self.ld3.deleteLater()*app = QtWidgets.QApplication(sys.argv)  
w = MainWindow()  
app.exec\_()

## Пример работы программы

В интерфейсе можно выбрать файл с точками t,x,y. Также можно выбрать каким способом можно ввести дельты (ползунком или полем), выбрать желаемую структуру.

Кнопка график осуществляет неавтоматический расчет дельт и передаточных. Автоматический ввод не требует ввода дельт.

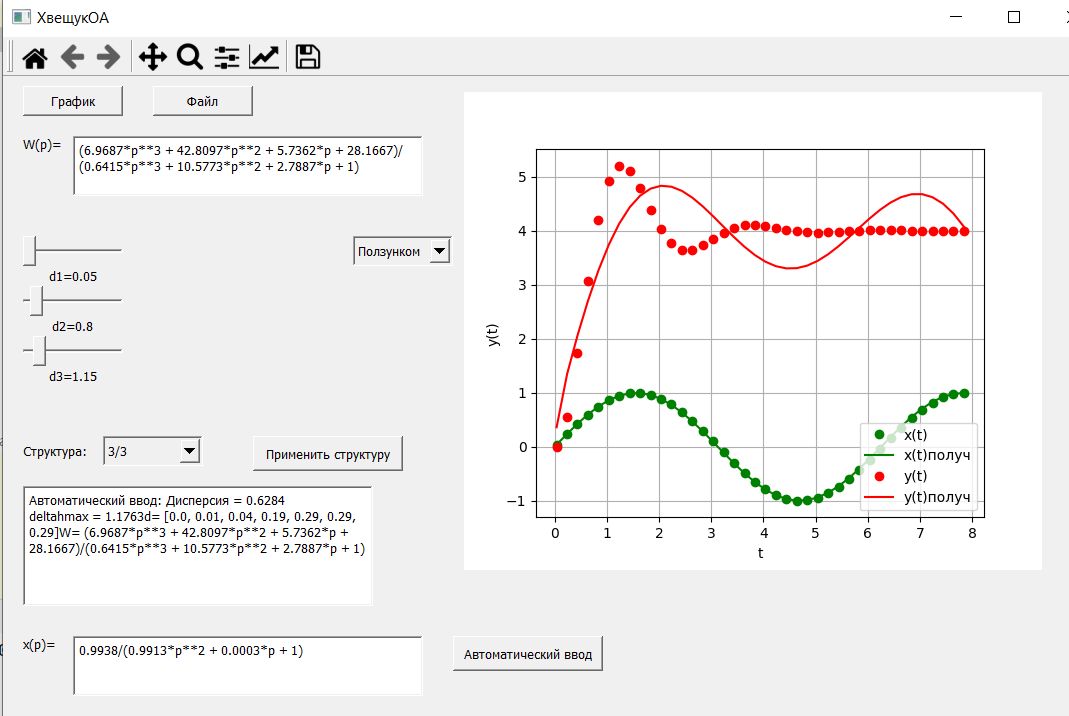


Рисунок 4 – Пример работы программы при воспроизведении входного воздействия.

# Задание №5

## Постановка задачи

Дана разомкнутая система с отрицательной обратной связью, в которой находится пропорциональное звено (коэффициент). По заданным критериям качества (перерегулирование, время регулирования, установившееся значение) и переходной характеристике объекта необходимо синтезировать регулятор, приводящий этот объект к желаемым характеристикам.

## Решение задачи

Передаточная функция замкнутой системы с отрицательной обратной связью, в которой находится пропорциональное звено выглядит следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  | (17) |

Отсюда получим зависимость передаточной функции регулятора от желаемой ПФ и передаточной функции объекта управления:

|  |  |  |
| --- | --- | --- |
|  |  | (18) |

Для того чтобы решить эту задачу необходимо найти передаточные функции желаемой системы и объекта управления.

Для нахождения желаемой передаточной функции необходимо расставить точки так, чтобы они соответствовали заданным критериям качества, и далее используя вещественный интерполяционный метод получить коэффициенты ПФ.

## Алгоритмы

Алгоритмы korni3(K) и korni2(K) представлены в 4 задании.

buildH(treg, hyst, reg) –формирует точки t и h по заданным параметрам

PF\_h(t,x, di) – расчитывает передаточную по точкам

Ksolve(d, t,x, n,hyst=0) – вычисляет коэффициенты передаточной, решая СЛАУ при определенной структуре

fp\_K(K,n) – получая структуру на вход выводит нужную формулу передаточной

automath(A,struct,hyst,top,amount) – при заданной структуре подбирает дельты для вычисления передаточной и hh. Дельты подбираются в циклах и выбираются те дельты, при которых дисперсия и максимальное отклонения минимальны

ParamH(t, h) – выводит нужные параметры переходной характеристики.

## Код

test1.py

**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**import** math  
**import** mpmath **as** mp*#осторожно,библиотеки могут конфликтовать***from** sympy **import** \**#осторожно,библиотеки могут конфликтовать***def** korni3(K):  
 K=[1,K[1]/K[0],K[2]/K[0],K[3]/K[0]]  
 a=K[1]  
 b=K[2]  
 c=K[3]  
 Q=(a\*\*2-3\*b)/9  
 R=(2\*a\*\*3-9\*a\*b+27\*c)/54  
 S=(Q\*\*3-R\*\*2).real  
 Q=Q.real  
 **if** S>0:  
 fi=(np.arccos(R/Q\*\*(1.5)))/3  
 x1=-2\*(Q)\*\*(0.5)\*np.cos(fi) - a/3  
 x2 = -2 \* (Q) \*\* (0.5) \* np.cos(fi+2\*np.pi/3) - a / 3  
 x3 = -2 \* (Q) \*\* (0.5) \* np.cos(fi - 2 \* np.pi / 3) - a / 3  
 **if** S < 0:  
 **if** Q>0:  
 fi=(np.arccosh(abs(R)/abs(Q)\*\*(1.5)))/3  
 x1= -2\*np.sign(R) \* abs(Q)\*\*(0.5) \* np.cosh(fi) - a / 3  
 x2 =complex(np.sign(R) \* abs(Q) \*\* (0.5)\* np.cosh(fi)- a / 3,(3\*abs(Q))\*\*(0.5)\*np.sinh(fi))  
 x3 =complex(np.sign(R) \* abs(Q) \*\* (0.5)\* np.cosh(fi)- a / 3,-(3\*abs(Q))\*\*(0.5)\*np.sinh(fi))  
 **if** Q < 0:  
 fi = (np.arcsinh(abs(R) / abs(Q) \*\* (1.5))) / 3  
 x1 = -2 \* np.sign(R) \* abs(Q) \*\* (0.5) \* np.sinh(fi) - a / 3  
 x2 = complex(np.sign(R) \* abs(Q) \*\* (0.5) \* np.sinh(fi) - a / 3, (3 \* abs(Q)) \*\* (0.5) \* np.cosh(fi))  
 x3 = complex(np.sign(R) \* abs(Q) \*\* (0.5) \* np.sinh(fi) - a / 3, -(3 \* abs(Q)) \*\* (0.5) \* np.cosh(fi))  
 **if** Q==0:  
 x1=-(c-a\*\*3/27)\*\*(1/3)-a/3  
 x2=complex(-(a+x1)\*0.5,0.5\*(abs((a-3\*x1)\*(a+x1)-4\*b)\*\*(0.5)))  
 x3 = complex(-(a + x1) \* 0.5, -0.5 \* (abs((a - 3 \* x1) \* (a + x1) - 4 \* b) \*\* (0.5)))  
 **if** S==0:  
 x1 = -2 \* (R) \*\* (1/3) - a / 3  
 x2 = (R) \*\* (1 / 3) - a / 3  
 x3 = (R) \*\* (1 / 3) - a / 3  
 **return** (x1,x2,x3)  
**def** korni2(K):  
 a = K[1]  
 b = K[2] *# b0/(a1d^2+a2d+1)* dis = b \*\* 2 - 4 \* a  
 **if** dis < 0:  
 x1 = complex(-b / (2 \* a), abs(dis) \*\* (0.5) / (2 \* a))  
 x2 = complex(-b / (2 \* a), -abs(dis) \*\* (0.5) / (2 \* a))  
 **else**:  
 x1 = (-b + abs(dis) \*\* (0.5)) / (2 \* a)  
 x2 = (-b - abs(dis) \*\* (0.5)) / (2 \* a)  
 **return** (x1,x2)  
**def** buildH(treg, hyst, reg):  
 hmax=hyst\*(1+reg\*0.01)  
 bot = 0.95 \* hyst  
 top = 1.05 \* hyst  
 regg = 0.85 \* reg \* 0.01  
 *#40  
 #коэффициенты умножения для времени* regg = 0.90 \* reg \* 0.01  
 hmin = hyst \* (1 - regg)  
 **if** reg>6:  
 k=np.array([0,0.75\*hyst,(hmax + hmin) / 2,hmax,(hmax + hmin) / 2,hmin,bot,top,hyst])  
 n=6  
 **elif** 5<=reg<=6:  
 hmin=bot  
 k = np.array([0, 0.75 \* hyst, (hmax + hmin) / 2, hmax, (hmax + hmin) / 2, hmin, hyst\*0.97, hyst\*1.03, hyst])  
 n=5  
 **elif** reg < 5:  
 k = np.array([0, 0.5 \* hyst,0.7 \* hyst,0.85 \* hyst,0.95 \* hyst, hyst,hmax,hmin, hyst])  
 n=4  
 A=[]  
 lenk=len(k)  
 **for** i **in** range (lenk):  
 A+=[[treg\*i/n,k[i]]]  
 A=np.array(A)  
 **return** A  
  
**def** PF\_h(t,x, di): *#вход по точкам* G=0  
 **for** i **in** range(len(t)):  
 **if** (i==0): dt=t[i+1]-t[i]  
 **else**:dt=t[i]-t[i-1]  
 G += x[i] \* np.exp(-di \* t[i]) \* dt  
 G=G\*di  
 **return** (G)  
**def** Ksolve(d, t,x, n,hyst=0): *# вычисление коэффициентов* Matr = []  
 b = []  
 **for** i **in** range(len(d)):  
 W = PF\_h(t,x, d[i])  
 **if** n == **'3/3'**:  
 Matr += [[d[i] \*\* 3, d[i] \*\* 2, d[i], 1, -W \* d[i] \*\* 3, -W \* d[i] \*\* 2,-W \* d[i]]]  
 **if** n == **'2/2'**:  
 Matr += [[d[i] \*\* 2, d[i], 1, -W \* d[i] \*\* 2, -W \* d[i]]]  
 **if** n == **'1/1'**:  
 Matr += [[d[i], 1, -W \* d[i]]] *# W=a0\*p+a1/(a2\*p+1)* **if** n == **'0/2'**:  
 Matr += [[-W \* d[i] \*\* 2, -W \* d[i]]]  
 b += [W - hyst]  
 **else**:  
 b += [W]  
 K = np.linalg.solve(Matr, b)  
 **if** n == **'0/2'**:  
 K = np.insert(K, 0, hyst)  
 **return** K  
**def** fp\_K(K,n):  
 **if** n == **'3/3'**:  
 fp = **lambda** p: (K[0] \* p \*\* 3 + K[1] \* p \*\* 2 + K[2] \* p + K[3]) / (  
 K[4] \* p \*\* 3 + K[5] \* p \*\* 2 + K[6] \* p + 1)  
 **if** n == **'2/2'**:  
 fp = **lambda** p: (K[0] \* p \*\* 2 + K[1] \* p + K[2]) / (K[3] \* p \*\* 2 + K[4] \* p + 1)  
 **if** n == **'0/2'**:  
 fp = **lambda** p: (K[0]) / (K[1] \* p \*\* 2 + K[2] \* p + 1)  
 **if** n == **'1/1'**:  
 fp = **lambda** p: (K[0] \* p + K[1]) / (K[0] \* p + 1)  
 **return** fp  
**def** reth(t,K,n,rbool=**False**,top=0,amount=0):  
 **if** n==**'3/3'**:  
 K1=[K[4],K[5],K[6],1]  
 x1,x2,x3=korni3(K1)  
 **if** n==**'1/1'**:*#a0\*p+a1/a2\*p+1* x1 = -1/K[2]  
 K1=[K[2]]  
 **if** n==**'2/2'**:  
 K1 = [K[3]]  
 x1, x2 = korni2([K[2],K[3],K[4]])  
 **if** n==**'0/2'**:  
 x1, x2 = korni2(K)  
 K1=[K[1]]  
 m=1/(K1[0])  
 h = []  
 **if** n == **'3/3'**:*#a0/a1p^2+a2\*p+1 \* a0x/a1xp^2+a2x\*p+1* A1 = (K[0] \* x1 \*\* 3 + K[1] \* x1 \*\* 2 + K[2] \* x1 + K[3])\*m / ((x1 - x2) \* (x1 - x3) \* x1)  
 A2 = (K[0] \* x2 \*\* 3 + K[1] \* x2 \*\* 2 + K[2] \* x2 + K[3]) \*m/ ((x2 - x1) \* (x2 - x3) \* x2)  
 A3 = (K[0] \* x3 \*\* 3 + K[1] \* x3 \*\* 2 + K[2] \* x3 + K[3])\*m / ((x3 - x2) \* (x3 - x1) \* x3)  
 A4 = K[3]\*m / (-x1 \* x2 \* x3)  
 **if** rbool == **True**:  
 **for** i **in** range(1, math.ceil(top \* amount)):  
 i = i / amount  
 h += [(A1 \* np.exp(x1 \* i) + A2 \* np.exp(x2 \* i) + A3 \* np.exp(x3 \* i) + A4).real]  
 **else**:  
 **for** i **in** t:  
 h += [(A1 \* np.exp(x1 \* i) + A2 \* np.exp(x2 \* i) + A3 \* np.exp(x3 \* i) + A4).real]  
 **if** n == **'1/1'**:  
 A1 = (K[0] \* x1 + K[1])\*m/ x1  
 A2 = -K[1]\*m / x1  
 **if** rbool == **True**:  
 **for** i **in** range(1, math.ceil(top \* amount)):  
 i = i / amount  
 h += [(A2 + A1 \* np.exp(x1 \* i)).real]  
 **else**:  
 **for** i **in** t:  
 h += [(A2 + A1 \* np.exp(x1 \* i)).real]  
 **if** n == **'0/2'**:  
  
 A1 = K[0]\*m / ((x1 - x2) \* x1)  
 A2 = K[0]\*m / ((x2 - x1) \* x2)  
 A3 = K[0]\*m / (x1 \* x2)  
 h = []  
 **if** rbool == **True**:  
 **for** i **in** range(1, math.ceil(top \* amount)):  
 i = i / amount  
 h += [(A3 + A1 \* np.exp(x1 \* i) + A2 \* np.exp(x2 \* i)).real]  
 **else**:  
 **for** i **in** t:  
 h += [(A3 + A1 \* np.exp(x1 \* i) + A2 \* np.exp(x2 \* i)).real]  
 **if** n == **'2/2'**:  
 A1 = (K[0] \* x1 \*\* 2 + K[1] \* x1 + K[2])\*m / ((x1 - x2) \* x1)  
 A2 = (K[0] \* x2 \*\* 2 + K[1] \* x2 + K[2]) \*m/ ((x2 - x1) \* x2)  
 A3 = K[2] \*m/ (x1 \* x2)  
 h = []  
 **if** rbool == **True**:  
 **for** i **in** range(1, math.ceil(top \* amount)):  
 i = i / amount  
 h += [(A3 + A1 \* np.exp(x1 \* i) + A2 \* np.exp(x2 \* i)).real]  
 **else**:  
 **for** i **in** t:  
 h += [(A3 + A1 \* np.exp(x1 \* i) + A2 \* np.exp(x2 \* i)).real]  
 **return** h  
**def** automath(A,struct,hyst,top,amount):  
 d = [0, 0]  
 t=A[:, 0]  
 x=A[:, 1]  
 lent = len(t)  
 minmax=100000  
 disp\_disp=100000  
 delta=[]  
 **if** struct == **'3/3'**:  
 d = [0, 0, 0, 0, 0,0,0]  
 **if** struct == **'2/2'**:  
 d = [0, 0, 0, 0, 0]  
 **if** struct == **'0/2' or** struct == **'1/1'**:  
 d = [0, 0, 0]  
 **if** struct == **'0/2'**:  
 m=10  
 **for** d[0] **in** range(1, 100, 1):  
 d[0] = d[0] / m  
 **for** d[1] **in** range(math.ceil(1 + d[0] \* m), 100, 1):  
 d[1] = d[1] / m  
 d=[d[0],d[1]]  
 K = Ksolve(d, t,x, struct,hyst)  
 **if not** np.all(abs(K) == K):  
 **continue** h= reth(t,K,struct)  
 disp=0  
 **for** i **in** range(lent):  
 disp += (x[i] - h[i]) \*\* 2  
 disp = (disp / (lent - 1)) \*\* (1 / 2)  
 maxn = max(abs(x - h))  
 **if** maxn<=minmax **and** disp<=disp\_disp:  
 minmax=maxn  
 disp\_disp=disp  
 delta=[d[0],d[1]]  
 print(delta, maxn)  
 **if** struct == **'3/3'**:  
 m=100  
 **for** d[0] **in** range(0, 1, 1):  
 d[0] = d[0] / m  
 **for** d[1] **in** range(math.ceil(1 + d[0] \* m), 5, 1):  
 d[1] = d[1] / m  
 **for** d[2] **in** range(math.ceil(1 + d[1] \* m), 10, 1):  
 d[2] = d[2] / m  
 **for** d[3] **in** range(math.ceil(1 + d[2] \* m), 15, 1):  
 d[3] = d[3] / m  
 **for** d[4] **in** range(math.ceil(1 + d[3] \* m), 20, 1):  
 d[4] = d[4] / m  
 **for** d[5] **in** range(math.ceil(1 + d[4] \* m), 25, 1):  
 d[5] = d[5] / m  
 **for** d[6] **in** range(math.ceil(1 + d[5] \* m), 30, 1):  
 d[6] = d[6] / m  
 d = [d[0], d[1], d[2], d[3], d[4],d[5],d[6]]  
 K = Ksolve(d, t, x, struct)  
 **if not** np.all(abs(K) == K):  
 **continue** h= reth(t, K, struct)  
 disp=0  
 **for** i **in** range(lent):  
 disp += (x[i] - h[i]) \*\* 2  
 disp = (disp / (lent - 1)) \*\* (1 / 2)  
 maxn = max(abs(x - h))  
 **if** maxn<=minmax **and** disp<=disp\_disp:  
 minmax=maxn  
 disp\_disp=disp  
 delta=[d[0], d[1], d[2], d[3], d[4],d[5],d[6]]  
 print(d,maxn)  
 **if** struct == **'2/2'**:  
 m=10  
 **for** d[0] **in** range(0, 5, 1):  
 d[0] = d[0]/m  
 **for** d[1] **in** range(math.ceil(1 + d[0]\*m), 10, 1):  
 d[1] = d[1]/m  
 **for** d[2] **in** range(math.ceil(1 + d[1]\*m), 15, 1):  
 d[2] = d[2]/m  
 **for** d[3] **in** range(math.ceil(1 + d[2]\*m), 20, 1):  
 d[3] = d[3]/m  
 **for** d[4] **in** range(math.ceil(1 + d[3]\*m),25, 1):  
 d[4] = d[4]/m  
 d = [d[0], d[1], d[2], d[3], d[4]]  
 K = Ksolve(d, t, x, struct)  
 **if not** np.all(abs(K) == K):  
 **continue** h = reth(t, K, struct)  
 disp = 0  
 **for** i **in** range(lent):  
 disp += (x[i] - h[i]) \*\* 2  
 disp = (disp / (lent - 1)) \*\* (1 / 2)  
 maxn = max(abs(x - h))  
 **if** maxn <= minmax **and** disp <= disp\_disp:  
 minmax = maxn  
 disp\_disp = disp  
 delta = [d[0], d[1], d[2], d[3], d[4]]  
 print(d, maxn)  
 **if** struct == **'1/1'**:  
 m=10  
 **for** d[0] **in** range(1, 6, 1):  
 d[0] = d[0] / m  
 **for** d[1] **in** range(math.ceil(1 + d[0] \* m), 20, 1):  
 d[1] = d[1] / m  
 **for** d[2] **in** range(math.ceil(1 + d[1] \* m), 1000, 1):  
 d[2] = d[2] / m  
 d = [d[0], d[1], d[2]]  
 K = Ksolve(d, t, x, struct)  
 **if not** np.all(abs(K) == K):  
 **continue** h = reth(t, K, struct)  
 disp = 0  
 **for** i **in** range(lent):  
 disp += (x[i] - h[i]) \*\* 2  
 disp = (disp / (lent - 1)) \*\* (1 / 2)  
 maxn = max(abs(x - h))  
 **if** maxn <= minmax **and** disp <= disp\_disp:  
 minmax = maxn  
 disp\_disp = disp  
 delta = [d[0], d[1], d[2]]  
 print(delta, maxn)  
 **if** delta==[]:  
 print(**"Решение не найдено"**)  
 **return** 0,0,0,0  
 K = Ksolve(delta, t, x, struct,hyst)  
 hh = reth(t, K, struct,**True**,top,amount)  
 *#Kx = np.around(Kx, decimals=4, out=None)* pf=fp\_K(K, struct)  
 maxn = np.around(maxn, decimals=4, out=**None**)  
 disp= np.around(disp, decimals=4, out=**None**)  
 **return** hh,pf,maxn,disp  
**def** ParamH(t, h):  
 hyst = h[-1]  
 top = hyst \* 1.05  
 bot = hyst \* 0.95  
 i = len(h) - 1  
 **while** h[i] > bot **and** h[i] <= top:  
 i -= 1  
 treg = t[i]  
 **return** treg, hyst

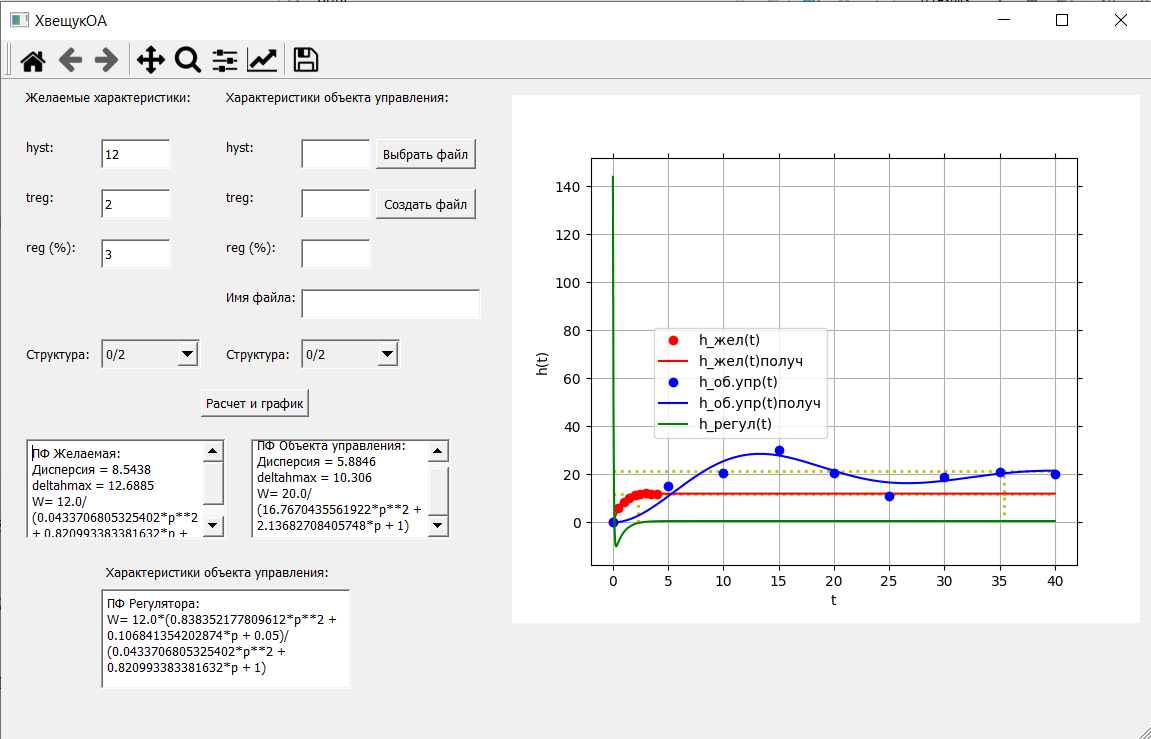
Main.py

**import** sys  
**from** PyQt5 **import** QtCore, QtWidgets  
**from** PyQt5.QtCore **import** Qt  
**from** PyQt5.QtWidgets **import** \*  
**from** matplotlib.backends.backend\_qt5agg **import** FigureCanvasQTAgg, NavigationToolbar2QT **as** NavigationToolbar  
**from** matplotlib.figure **import** Figure  
**import** numpy **as** np  
  
**from** mplwidget **import** MplWidget  
**from** test1 **import** \*  
  
**class** MplCanvas(FigureCanvasQTAgg):  
  
 **def** \_\_init\_\_(self, parent=**None**, width=5, height=4, dpi=100):  
 fig = Figure(figsize=(width, height), dpi=dpi)  
 self.axes = fig.add\_subplot(111)  
 super(MplCanvas, self).\_\_init\_\_(fig)  
  
  
**class** MainWindow(QtWidgets.QMainWindow):  
  
 **def** \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(MainWindow, self).\_\_init\_\_(\*args, \*\*kwargs)self.initUI()  
  
 **def** initUI(self):  
 self.setObjectName(**"MainWindow"**)  
 self.resize(1150, 700)  
 self.centralwidget = QtWidgets.QWidget(self)  
 self.centralwidget.setObjectName(**"centralwidget"**)  
 self.MplWidget = MplWidget(self.centralwidget)  
  
 self.MplWidget.setGeometry(QtCore.QRect(500, 5, 650, 550))  
 self.MplWidget.setMinimumSize(QtCore.QSize(100, 350))  
 self.MplWidget.setObjectName(**"MplWidget"**)  
 self.setCentralWidget(self.centralwidget)  
 self.addToolBar(NavigationToolbar(self.MplWidget.canvas, self))  
  
 self.l\_w = QLabel(self)  
 self.l\_w.move(25, 50)  
 self.l\_w.setText(**'Желаемые характеристики:'**)  
 self.l\_w.adjustSize()  
  
 self.lw\_h = QLabel(**'hyst:'**, self)  
 self.lw\_h.move(25, 100)self.lw\_h.adjustSize()  
  
 self.combo\_w\_h = QLineEdit(self)  
 self.combo\_w\_h.move(100, 100)  
 self.combo\_w\_h.resize(70, 30)  
  
 self.lw\_treg = QLabel(**'treg:'**, self)  
 self.lw\_treg.move(25, 150)  
 self.lw\_treg.adjustSize()  
  
 self.combo\_w\_t = QLineEdit(self)  
 self.combo\_w\_t.move(100, 150)  
 self.combo\_w\_t.resize(70, 30)  
  
 self.lw\_reg = QLabel(**'reg (%):'**, self)  
 self.lw\_reg.move(25, 200)  
 self.lw\_reg.adjustSize()  
  
 self.combo\_w\_r = QLineEdit(self)  
 self.combo\_w\_r.move(100, 200)  
 self.combo\_w\_r.resize(70, 30)  
  
 self.l\_w\_struct = QLabel(self)  
 self.l\_w\_struct.move(25, 300)  
 self.l\_w\_struct.setText(**'Структура:'**)  
  
 self.combo\_w\_struct = QComboBox(self)  
 self.combo\_w\_struct.addItem(**'3/3'**)  
 self.combo\_w\_struct.addItem(**'2/2'**)  
 self.combo\_w\_struct.addItem(**'1/1'**)  
 self.combo\_w\_struct.addItem(**'0/2'**)  
 self.combo\_w\_struct.move(100, 300)  
  
 self.lwant\_otv = QTextEdit(self)  
 self.lwant\_otv.move(25, 400)  
 self.lwant\_otv.resize(200, 100)  
  
 self.l\_o = QLabel(self)  
 self.l\_o.move(225, 50)  
 self.l\_o.setText(**'Характеристики объекта управления:'**)  
 self.l\_o.adjustSize()  
  
 self.lo\_h = QLabel(**'hyst:'**, self)  
 self.lo\_h.move(225, 100)  
 *# self.lo\_h.setText('W(p)=')* self.lo\_h.adjustSize()  
  
 self.combo\_o\_h = QLineEdit(self)  
 self.combo\_o\_h.move(300, 100)  
 self.combo\_o\_h.resize(70, 30)  
  
 self.lo\_treg = QLabel(**'treg:'**, self)  
 self.lo\_treg.move(225, 150)  
 self.lo\_treg.adjustSize()  
  
 self.combo\_o\_t = QLineEdit(self)  
 self.combo\_o\_t.move(300, 150)  
 self.combo\_o\_t.resize(70, 30)  
  
 self.lo\_reg = QLabel(**'reg (%):'**, self)  
 self.lo\_reg.move(225, 200)  
 self.lo\_reg.adjustSize()  
  
 self.combo\_o\_r = QLineEdit(self)  
 self.combo\_o\_r.move(300, 200)  
 self.combo\_o\_r.resize(70, 30)  
  
 self.lo\_namefile= QLabel(**'Имя файла:'**, self)  
 self.lo\_namefile.move(225, 250)  
 self.lo\_namefile.adjustSize()  
  
 self.combo\_o\_f = QLineEdit(self)  
 self.combo\_o\_f.move(300, 250)  
 self.combo\_o\_f.resize(180, 30)  
  
 self.l\_o\_struct = QLabel(self)  
 self.l\_o\_struct.move(225, 300)  
 self.l\_o\_struct.setText(**'Структура:'**)  
  
 self.combo\_o\_struct = QComboBox(self)  
 self.combo\_o\_struct.addItem(**'3/3'**)  
 self.combo\_o\_struct.addItem(**'2/2'**)  
 self.combo\_o\_struct.addItem(**'1/1'**)  
 self.combo\_o\_struct.addItem(**'0/2'**)  
 self.combo\_o\_struct.move(300, 300)  
  
 self.lobj\_otv = QTextEdit(self)  
 self.lobj\_otv.move(250, 400)  
 self.lobj\_otv.resize(200, 100)  
  
 self.l\_regul = QLabel(self)  
 self.l\_regul.move(105, 525)  
 self.l\_regul.setText(**'Характеристики объекта управления:'**)  
 self.l\_regul.adjustSize()  
  
 self.lregula\_otv = QTextEdit(self)  
 self.lregula\_otv.move(100, 550)  
 self.lregula\_otv.resize(250, 100)**global** filename  
 filename = **'FileObject'** *# добавлене кнопок* btn1 = QPushButton(**"Расчет и график"**, self)  
 btn1.move(200, 350)  
 btn1.adjustSize()  
 btn1.clicked.connect(self.update\_graph)  
  
 btn2 = QPushButton(**"Выбрать файл"**, self)  
 btn2.move(375, 100)  
 btn2.clicked.connect(self.getFileName)  
  
 btn3 = QPushButton(**"Cоздать файл"**, self)  
 btn3.move(375, 150)  
 *#btn3.setGeometry(800, 600, 200, 45)* btn3.clicked.connect(self.create\_obj)  
 *# если нажали на кнопку выполнить функции* self.statusBar()  
 self.setWindowTitle(**'ХвещукОА'**)  
 self.show()  
 **def** getFileName(self):  
 **global** filename  
 filename, filetype = QFileDialog.getOpenFileName(self, **"All Files(\*.\*)"**)  
 *#filename=filename[-9:]* **def** update\_graph(self):  
  
 self.lwant\_otv.clear()  
 self.lobj\_otv.clear()  
 self.lregula\_otv.clear()  
 self.MplWidget.canvas.axes.clear()  
**try**:  
 hyst = float(self.combo\_w\_h.text())  
 treg = float(self.combo\_w\_t.text())  
 reg = float(self.combo\_w\_r.text())  
 **except**:  
 QMessageBox.critical(self, **"Ошибка "**, **"Данные не введены, либо некорректны"**, QMessageBox.Ok)  
 **else**:  
 **if** hyst < 0 **or** treg < 0 **or** reg < 0:  
 QMessageBox.critical(self, **"Ошибка "**, **"параметры должны быть >0"**, QMessageBox.Ok)  
 **if** reg > 100:  
 QMessageBox.critical(self, **"Ошибка "**, **"регулирование должно быть до 100 %"**, QMessageBox.Ok)  
 **try**:  
 obj\_H = np.loadtxt(filename, delimiter=**' '**)  
 m,n=obj\_H.shape  
 **if** m < 5 **or not** n == 2:  
 QMessageBox.critical(self, **"Ошибка "**, **"Неправильная форма массива в файле"**, QMessageBox.Ok)  
 **return  
 if** len(obj\_H)>len(set(obj\_H[:,0])):  
 QMessageBox.critical(self, **"Ошибка "**, **"Некорректные данные в файле"**, QMessageBox.Ok)  
 **return  
 except**:  
 QMessageBox.critical(self, **"Ошибка "**, **"Некорректные данные в файле"**, QMessageBox.Ok)  
 **else**:  
 want\_H = buildH(treg, hyst, reg)  
 top = max(want\_H[-1, 0], obj\_H[-1, 0])  
 amount = 40  
  
 p = symbols(**'p'**)  
 struct\_want = self.combo\_w\_struct.currentText()  
 want\_HH, want\_pf, deltahmax, sigma = automath(want\_H, struct\_want, hyst, top, amount)  
 **if** [want\_HH, want\_pf, deltahmax, sigma]==[0,0,0,0]:  
 QMessageBox.critical(self, **"Ошибка "**, **"Нет стабильных решений для ПФ желаемой"  
 " при данной структуре"**, QMessageBox.Ok)  
 **return** struct\_obj = self.combo\_o\_struct.currentText()  
 obj\_HH, obj\_pf, deltahobj, sigmobj = automath(obj\_H, struct\_obj, obj\_H[-1, 1], top, amount)  
 **if** [obj\_HH, obj\_pf, deltahobj, sigmobj] == [0, 0, 0, 0]:  
 QMessageBox.critical(self, **"Ошибка "**, **"Нет стабильных решений для ПФ объекта управления"  
 " при данной структуре"**, QMessageBox.Ok)  
 **return** want\_otv = **'ПФ Желаемая:\n'** + **'Дисперсия = '** + str(sigma) + **'\ndeltahmax = '** + str(deltahmax) + \  
 **'\nW= '** + str(want\_pf(p))  
 obj\_otv = **'ПФ Объекта управления:\n'** + **'Дисперсия = '** + str(sigmobj) + **'\ndeltahmax = '** \  
 + str(deltahobj) + **'\nW= '** + str(obj\_pf(p))  
  
 regula\_pfh = **lambda** p: want\_pf(p) / obj\_pf(p) / p  
 regula\_otv = **'ПФ Регулятора:\n'** + **'W= '** + str(regula\_pfh(p) \* p)  
 regula\_h = []  
 t = []  
 **for** i **in** range(1, math.ceil(top \* amount)):  
 t += [i / amount]  
 regula\_h += [float(mp.invertlaplace(regula\_pfh, t[i - 1], method=**'talbot'**))]  
 self.MplWidget.canvas.axes.plot(want\_H[:, 0], want\_H[:, 1], **'ro'**,label=**'h\_жел(t)'**)  
 self.MplWidget.canvas.axes.plot(t, want\_HH, **'r'**,label=**'h\_жел(t)получ'**)  
 self.MplWidget.canvas.axes.plot(obj\_H[:, 0], obj\_H[:, 1], **'bo'**,label=**'h\_об.упр(t)'**)  
 self.MplWidget.canvas.axes.plot(t, obj\_HH, **'b'**,label=**'h\_об.упр(t)получ'**)  
 self.MplWidget.canvas.axes.plot(t, regula\_h, **'g'**,label=**'h\_регул(t)'**)  
 self.MplWidget.canvas.axes.legend(bbox\_to\_anchor=(0.5, 0.6))*#loc='lower right')* treg\_w, hyst\_w=ParamH(t, want\_HH)  
 treg\_o, hyst\_o = ParamH(t, obj\_HH)  
 self.MplWidget.canvas.axes.hlines(hyst\_w, 0, t[-1], color=**'y'**, linewidth=2, linestyle=**':'**)  
 self.MplWidget.canvas.axes.hlines(hyst\_o, 0, t[-1], color=**'y'**, linewidth=2, linestyle=**':'**)  
  
 self.MplWidget.canvas.axes.vlines(treg\_w, 0, hyst\_w, color=**'y'**, linewidth=2, linestyle=**':'**)  
 self.MplWidget.canvas.axes.vlines(treg\_o, 0, hyst\_o, color=**'y'**, linewidth=2, linestyle=**':'**)  
  
 self.MplWidget.canvas.axes.set\_xlabel(**'t'**)  
 self.MplWidget.canvas.axes.set\_ylabel(**'h(t)'**)  
 self.MplWidget.canvas.axes.grid()  
 self.MplWidget.canvas.draw()  
  
 self.lwant\_otv.setText(want\_otv)  
 self.lobj\_otv.setText(obj\_otv)  
 self.lregula\_otv.setText(regula\_otv)**def** create\_obj(self):self.MplWidget.canvas.axes.clear()  
 **try**:  
 hyst = float(self.combo\_o\_h.text())  
 treg = float(self.combo\_o\_t.text())  
 reg = float(self.combo\_o\_r.text())  
 file = self.combo\_o\_f.text()  
 **if** hyst < 0 **or** treg < 0 **or** reg < 0:  
 QMessageBox.critical(self, **"Ошибка "**, **"параметры должны быть >0"**, QMessageBox.Ok)  
 **elif** reg > 100:  
 QMessageBox.critical(self, **"Ошибка "**, **"регулирование должно быть до 100 %"**, QMessageBox.Ok)  
 **else**:  
 obj\_H = buildH(treg, hyst, reg)  
 np.savetxt(file, obj\_H)  
 **except**:  
 QMessageBox.critical(self, **"Ошибка "**, **"параметры объекта управления для создания файла не "  
 "определены"**, QMessageBox.Ok)  
app = QtWidgets.QApplication(sys.argv)  
w = MainWindow()  
app.exec\_()

## Пример работы программы

Для характеристики объекта управления можно выбрать файл с точками, также его можно создать если ввести для него параметры и имя файла.

Для расчетов ПФ Желаемой и ПФ объекта управления предусмотрен выбор структур, однако не при всех структурах можно найти решения, при которых коэффициенты положительные, при таких коэффициентах графики не выводятся.



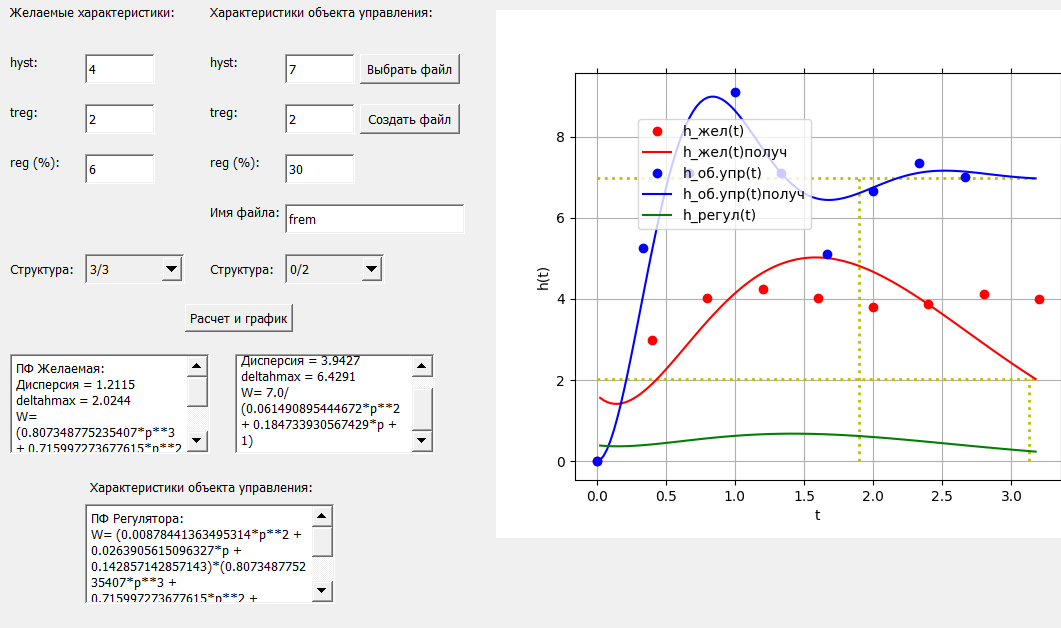


Рисунок 5 – Пример работы программы.

# Задание №6

## Постановка задачи

Программа должна считывать матричное представление сети (произвольной размерности с произвольными связями и функциями активации), после чего отправляет на вход нейронной сети вектор соответствующих размеров и вычисляет выходной вектор.

## Решение задачи

Простую однородную нейронную сеть можно представить следующим образом:

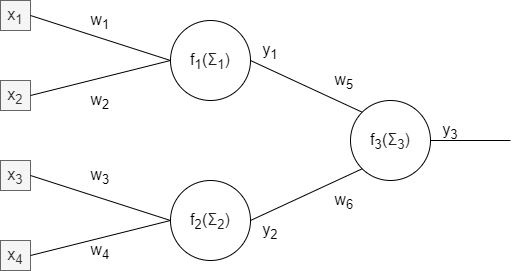


Рисунок 6.1 – Однородная нейронная сеть

Здесь входы X подаются на нейроны первого слоя, при этом перемножаются на веса W, суммируются и проходят функцию активации F. Таким образом на выходе нейрона формируется выход Y. Этот выход будет является входом для следующего слоя нейронов.

Матрицу весов такой сети можно представить как:

В 0 строке обозначены номера слоев.

Пусть

Тогда выход 1 слоя будет равен

Или проще говоря это

Аналогично выход второго слоя

A1 и A2 обозначают матрицы 1 и 2 слоя соответственно.

AmountNeiron – определяет количество нейронов в каждом слое по порядку, включая входы.

forward – расчитывает выход каждого слоя. Act – массив функций активации

Данные функции могут использоваться при любой однородной НС

## Код

Test1.py

**import** numpy **as** np  
**def** forward(act,X,A):*#прямое распространение - расчет выходов* f = **lambda** x,t: eval(str(t))  
 n=AmountNeiron(X,A)  
 t=0  
 out=X  
 M=[X]  
 **for** i **in** range(len(n)-1):  
 A1=A[1:n[i] + 1,t:n[i+1]+t]  
 sum=np.dot(out,A1)  
 out=[f(sum[i],act[i]) **for** i **in** range(len(sum))]  
 t+=len(A1[0,:])  
 M+=[out]  
 **return** M  
**def** AmountNeiron(X,A):*#считает количество нейронов в слое* lensloi = int(A[0, -1])  
 n = [len(X)]  
 **for** sloi **in** range(1,lensloi+1):  
 n += [list(A[0, :]).count(sloi)] *# количество нейронов в слое  
 #n.insert(0, len(X))* **return** n

Main.py

**import** sys  
**from** PyQt5 **import** QtCore, QtWidgets  
**from** PyQt5.QtCore **import** Qt  
**from** PyQt5 **import** QtGui  
**from** PyQt5.QtWidgets **import** \*  
**from** functools **import** partial  
  
**from** test1 **import** \*  
  
**class** MainWindow(QtWidgets.QMainWindow):  
  
 **def** \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(MainWindow, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.initUI()  
  
 **def** initUI(self):  
  
 self.setObjectName(**"MainWindow"**)  
 self.resize(1000, 600)  
 self.centralwidget = QtWidgets.QWidget(self)  
 self.centralwidget.setObjectName(**"centralwidget"**)  
 self.setCentralWidget(self.centralwidget)  
 self.lwant\_otv = QTextEdit(self)  
 self.lwant\_otv.move(275, 25)  
 self.lwant\_otv.resize(500, 200)  
 font = QtGui.QFont()  
 font.setPointSize(13)  
 self.lwant\_otv.setFont(font)  
 **global** FileStruct,FileActivation,FileX  
 FileStruct= **'Structure NN'** FileActivation = **'activation'** FileX = **'X'** *# добавлене кнопок* btn1 = QPushButton(**"Расчет"**, self)  
 btn1.move(25, 25)  
 btn1.adjustSize()  
 btn1.clicked.connect(self.update\_graph)  
  
 btn2 = QPushButton(**"Выбрать файл для структуры НС"**, self)  
 btn2.move(25, 75)  
 btn2.adjustSize()  
 btn2.clicked.connect(partial(self.getFileName,FileStruct))  
 btn4 = QPushButton(**"Выбрать файл для расчета выхода"**, self)  
 btn4.move(25, 125)  
 btn4.adjustSize()  
 btn4.clicked.connect(partial(self.getFileName,FileX))  
  
 btn5 = QPushButton(**"Выбрать файл для функций активации"**, self)  
 btn5.move(25, 175)  
 btn5.adjustSize()  
 btn5.clicked.connect(partial(self.getFileName,FileActivation))  
 *# если нажали на кнопку выполнить функции* self.statusBar()  
 self.setWindowTitle(**'ХвещукОА'**)  
 self.show()  
 **def** getFileName(self,filename):  
 filename, filetype = QFileDialog.getOpenFileName(self, **"All Files(\*.\*)"**)  
 *#filename=filename[-9:]* **def** update\_graph(self):  
 **try**:  
 act = np.loadtxt(FileActivation, dtype=str)  
 A = np.loadtxt(FileStruct)  
 X = np.loadtxt(FileX)  
 **if** len(A)!=len(X)+1:  
 QMessageBox.critical(self, **"Ошибка "**, **"Выходных нейронов больше чем выходов"**, QMessageBox.Ok)  
 **return  
 except**:  
 QMessageBox.critical(self, **"Ошибка "**, **"Некорректные данные в файлах"**, QMessageBox.Ok)  
 **return** M=forward(act, X, A)  
 self.lwant\_otv.setText(**'Входы: '**+str(M[:][0])+**'\nВыходы скрытых слоев: '**+str(M[:][1:-1])+**'\nВыходы: '**+str(M[:][-1]))  
app = QtWidgets.QApplication(sys.argv)  
w = MainWindow()  
app.exec\_()

## Пример работы программы

Выбираются файлы со структурой НС, одномерный массив точек входов, и файл с функциями активации.

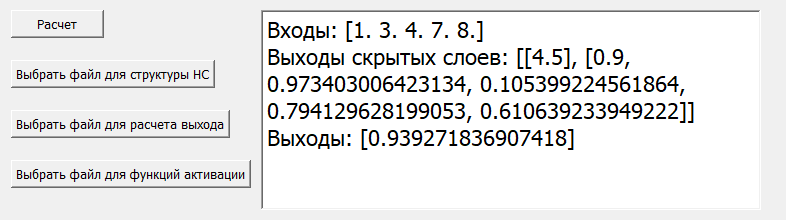


Рисунок 6.2 – Пример работы программы.

# Задание №7

## Постановка задачи

Необходимо написать программу для обучения нейронной сети конфигурации «много входов – много выходов».

## Решение задачи

Для обучения нейронной сети было принято решение использовать метод обратного распространения ошибки. Его суть заключается в том, что на сеть подается входной вектор, вычисляется выходной и сравнивается с желаемым выходным вектором. На основе этого сравнения вычисляется ошибка для выходного слоя нейронов. Далее на основе этой ошибки и промежуточных векторов вычисляется ошибка для предыдущего слоя, и так до самого первого, то есть в направлении обратном прямому распространению сигналов в обычном режиме работы. Отсюда и название метода.

Была выбрана двухслойная структура сети, в которой каждый нейрон связан с каждым входом.

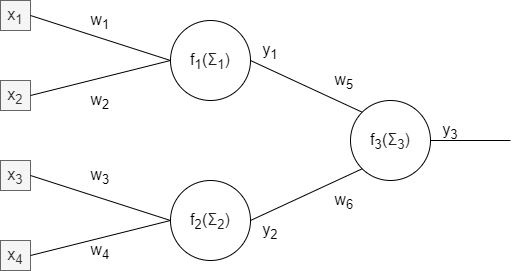


Рисунок 7.1 – Однородная нейронная сеть

Сначала идет инициализация весов значениями от -0.5 до 0.5. Чем меньше начальные веса тем медленнее сеть перенасыщается далее расчитывается выход, вычисляется ошибка сети

Далее умножается на производную функции активации, получается градиент

Выбирая скорость обучения a

На первой эпохе все

Переходя к следующему слою (1)

Если выходов много, то

где i – нейроны на слое, а k – нейроны на следующем слое связывает i-ый нейрон с k-ым.

Так заканчивается 1 эпоха.

Далее будет представлена реализация для обучения только с 2-мя слоями сети.

Также стоит отметить, что функция активации гиперболический тангенс дает и отрицательные, и положительные ответы, а логистическая только неотрицательные.

## Код

Test1.py

**import** numpy **as** np  
**from** sympy **import** \*  
**import** matplotlib.pyplot **as** plt  
*#from main import f***def** dif(t,f):*#считает производную* x = symbols(**'x'**)  
 ff=f(x)  
 m=diff(ff)  
 dif=**lambda** x: eval(str(m))  
 **return** dif(t)  
**def** AmountNeiron(X,A):*#считает количество нейронов в слое* lensloi = int(A[0, -1])  
 n = [len(X)]  
 **for** sloi **in** range(1,lensloi+1):  
 n += [list(A[0, :]).count(sloi)] *# количество нейронов в слое  
 #n.insert(0, len(X))* **return** n  
*#f = lambda x: eval(str(act))***def** forward(f,X,A):*#прямое распространение - расчет выходов* n=AmountNeiron(X,A)  
 A2 = A[1:n[1] + 1, -n[2]:]  
 A1 = A[1:n[0] + 1, :n[1]]  
 sum=np.dot(X,A1)  
 out1=[f(x) **for** x **in** sum]  
 sum = np.dot(out1, A2)  
 out2=[f(x) **for** x **in** sum]  
 M=[out1,out2]  
 **return** M  
def RandomWeight(n,m,lenm):  
 t=0  
 if n[0]>=n[1]:  
 t=n[0]  
 else:  
 t=n[1]  
 A=np.zeros((1+t,n[1]+n[2]))  
 r=[1 for i in range (n[1])]  
 r+=[2 for i in range (n[2])]  
 A[0,:]=r  
 A2=np.random.randint(-m, m, size=(n[1],n[2]))/lenm  
 A1 = np.random.randint(-m,m, size=(n[0], n[1]))/lenm  
 A[1:n[1] + 1, -n[2]:]=A2  
 A[1:n[0] + 1, :n[1]]=A1  
 return A

**def** backprop(f,A,XY,n,a):  
 x = symbols(**'x'**)  
 df=**lambda** x: dif(x,f)  
 N=AmountNeiron(XY[:,0],A)  
 n0=N[0]  
 n1=N[1]  
 n2=N[2]  
 A2=A[1:n1+1,-n2:]  
 A1=A[1:n0+1,:n1]  
 dW2=[[0 **for** i **in** range (n2)] **for** j **in** range (n1)]  
 dW1 = [[0 **for** i **in** range (n1)] **for** j **in** range (n0)]  
 **for** r **in** range (n):  
 **for** t **in** range (int(len(XY[0,:])/2)):  
 out=XY[:n0,2\*t]  
 M= forward(f,out,A)  
 out2 = np.array(M[:][1])  
 out1 = np.array(M[:][0])  
 y=XY[:, 2 \* t + 1]  
 e=(y - out2)*#ошибка* d2=[]  
 **for** k **in** range(n2):  
 d2+=[e[k]\*df(out2[k])]  
 print(**'d2'**,d2)  
 D1=(A2).dot(d2)  
 d1=[]  
 **for** j **in** range(n1):  
 d1+=[df(out1[j])\*D1[j]]  
 **for** j **in** range (n1):  
 **for** k **in** range (n2):  
 dW2[j][k]=a\*dW2[j][k]-(1 - a) \* out1[j] \* d2[k]  
 *#A2[j][k] = A2[j][k] - a \* out1[j] \* d2[k]* **for** i **in** range(n0):  
 **for** j **in** range(n1):  
 dW1[i][j]=a\*dW1[i][j]-(1 - a) \* out[i] \* d1[j]  
 *#A1[i][j] = A1[i][j] - a \* out[i] \* d1[j]* A2=A2-dW2  
 A1=A1-dW1  
 A[1:n1 + 1, -n2:]=A2  
 A[1:n0 + 1,:n1]=A1  
 **return** A

Main.py

**import** sys  
**from** PyQt5 **import** QtCore, QtWidgets  
**from** PyQt5.QtCore **import** Qt  
**from** PyQt5.QtWidgets **import** \*  
**from** matplotlib.backends.backend\_qt5agg **import** FigureCanvasQTAgg, NavigationToolbar2QT **as** NavigationToolbar  
**from** matplotlib.figure **import** Figure  
**import** numpy **as** np  
**from** functools **import** partial  
  
**from** mplwidget **import** MplWidget  
**from** test1 **import** \*  
  
**class** MplCanvas(FigureCanvasQTAgg):  
  
 **def** \_\_init\_\_(self, parent=**None**, width=5, height=4, dpi=100):  
 fig = Figure(figsize=(width, height), dpi=dpi)  
 self.axes = fig.add\_subplot(111)  
 super(MplCanvas, self).\_\_init\_\_(fig)  
  
  
**class** MainWindow(QtWidgets.QMainWindow):  
  
 **def** \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(MainWindow, self).\_\_init\_\_(\*args, \*\*kwargs)  
 sc = MplCanvas(self, width=5, height=4, dpi=100)  
 *#df = pd.DataFrame([[0, 10], [5, 15], [2, 20], [15, 25], [4, 10],], columns=['A', 'B'])  
  
 #df.plot(ax=sc.axes)* self.initUI()  
  
 **def** initUI(self):  
 self.setObjectName(**"MainWindow"**)  
 self.resize(1000, 600)  
 self.centralwidget = QtWidgets.QWidget(self)  
 self.centralwidget.setObjectName(**"centralwidget"**)  
 self.MplWidget = MplWidget(self.centralwidget)  
  
 self.MplWidget.setGeometry(QtCore.QRect(300, 5, 650, 550))  
 self.MplWidget.setMinimumSize(QtCore.QSize(100, 350))  
 self.MplWidget.setObjectName(**"MplWidget"**)  
 self.setCentralWidget(self.centralwidget)  
 self.addToolBar(NavigationToolbar(self.MplWidget.canvas, self))  
  
 self.lo\_h = QLabel(**'Количество эпох:'**, self)  
 self.lo\_h.move(25, 75)  
 *# self.lo\_h.setText('W(p)=')* self.lo\_h.adjustSize()  
 self.combo\_o\_h = QLineEdit(self)  
 self.combo\_o\_h.move(150, 75)  
 self.combo\_o\_h.resize(70, 30)  
 **global** FileStruct,FileTestXY,FileActivation,FileXY  
 FileStruct= **'Structure NN'** FileTestXY=**'TestXY'** FileActivation = **'activation'** FileXY = **'XY'** *# добавлене кнопок* btn1 = QPushButton(**"Расчет и график"**, self)  
 btn1.move(25, 100)  
 btn1.adjustSize()  
 btn1.clicked.connect(self.update\_graph)  
  
 btn2 = QPushButton(**"Выбрать файл для структуры НС"**, self)  
 btn2.move(25, 150)  
 btn2.adjustSize()  
 btn2.clicked.connect(partial(self.getFileName,FileStruct))  
 btn3 = QPushButton(**"Выбрать файл для точек"**, self)  
 btn3.move(25, 200)  
 btn3.adjustSize()  
 btn3.clicked.connect(partial(self.getFileName,FileTestXY))  
 btn4 = QPushButton(**"Выбрать файл для расчета выхода"**, self)  
 btn4.move(25, 250)  
 btn4.adjustSize()  
 btn4.clicked.connect(partial(self.getFileName,FileXY))  
  
 btn5 = QPushButton(**"Выбрать файл для функции активации"**, self)  
 btn5.move(25, 300)  
 btn5.adjustSize()  
 btn5.clicked.connect(partial(self.getFileName,FileActivation))  
 *# если нажали на кнопку выполнить функции* self.statusBar()  
 self.setWindowTitle(**'ХвещукОА'**)  
 self.show()  
 **def** getFileName(self,filename):  
 filename, filetype = QFileDialog.getOpenFileName(self, **"All Files(\*.\*)"**)  
 *#filename=filename[-9:]* **def** update\_graph(self):  
 self.MplWidget.canvas.axes.clear()  
 **try**:  
 epoch = int(self.combo\_o\_h.text())*#чтение количества итераций* **except**:  
 QMessageBox.critical(self, **"Ошибка "**, **"Данные не введены, либо некорректны"**, QMessageBox.Ok)  
 **return  
 if** epoch < 1 **or** epoch >10000:  
 QMessageBox.critical(self, **"Ошибка "**, **"Количество эпох должно быть в пределах от 1 до 10000"**, QMessageBox.Ok)  
 **return  
 try**:  
 act = np.loadtxt(FileActivation, dtype=str)  
 A = np.loadtxt(FileStruct)  
 TestXY = np.loadtxt(FileTestXY)  
 XY = np.loadtxt(FileXY)  
 **if** len(A)!=len(XY)+1 **or** len(A)!=len(TestXY)+1:  
 QMessageBox.critical(self, **"Ошибка "**, **"Выходных нейронов больше чем выходов"**, QMessageBox.Ok)  
 **return  
 except**:  
 QMessageBox.critical(self, **"Ошибка "**, **"Некорректные данные в файлах"**, QMessageBox.Ok)  
 **return** f = **lambda** x: eval(str(act))  
 n=AmountNeiron(XY[:,0],A)  
 A = RandomWeight(n, 1, 100)  
 A1=backprop(f,A,TestXY,epoch,0.1)  
 M= forward(f,XY[:,0],A1)  
 self.MplWidget.canvas.axes.plot(XY[:,0], **'r'**, label=**'вход'**)  
 self.MplWidget.canvas.axes.plot(XY[:,1], **'ro'**, label=**'выход'**)  
 self.MplWidget.canvas.axes.plot(M[:][1], **'bo'**, label=**'выход по НС'**)  
 self.MplWidget.canvas.axes.legend(bbox\_to\_anchor=(0.5, 0.6)) *# loc='lower right')* self.MplWidget.canvas.axes.grid()  
 self.MplWidget.canvas.draw()  
app = QtWidgets.QApplication(sys.argv)  
w = MainWindow()  
app.exec\_()

## Пример работы программы

Есть возможность вводить количество эпох, выбирать начальный файл структуры нейроной сети (который в последствии будет заменен случайными весами), выборки для обучения (файл для точек), для каких входов расчитать выход (должен содержать вход и выход для проверки), и файл для функции активации (должен содержать 1 функцию). Далее будет представлен пример работы сети на логистической функции.

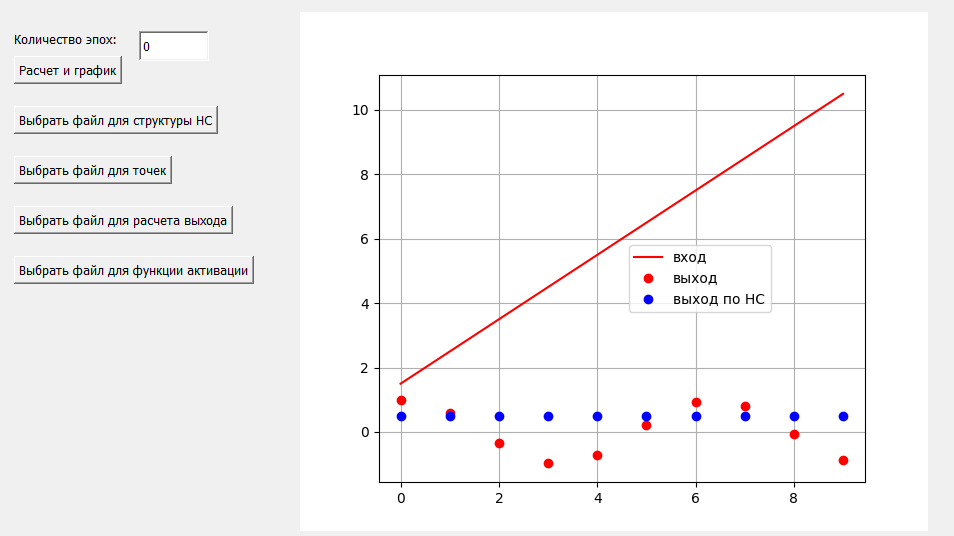


Рисунок 7.2– Сеть не обучена.

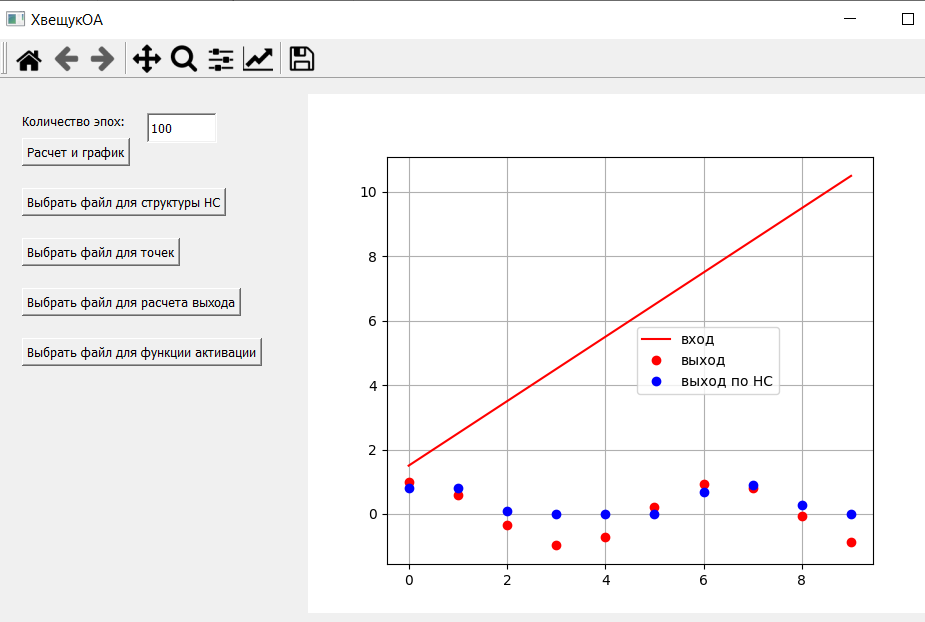


Рисунок 7.3 – сеть после 100 эпох

Приложение

Этот файл используется во всех заданиях, кроме 6:

mplwidget.py

from PyQt5.QtWidgets import \*  
from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg  
from matplotlib.figure import Figure  
  
class MplWidget(QWidget):  
   
 def \_\_init\_\_(self, parent=None):  
 QWidget.\_\_init\_\_(self, parent)  
  
 self.canvas = FigureCanvasQTAgg(Figure())  
  
 vertical\_layout = QVBoxLayout()  
 vertical\_layout.addWidget(self.canvas)  
  
 self.canvas.axes = self.canvas.figure.add\_subplot(111)  
 self.setLayout(vertical\_layout)