Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Контрольная работа

по дисциплине  «Моделирование»

Вариант 26

Студент гр. 600541 Зюзькин А.А

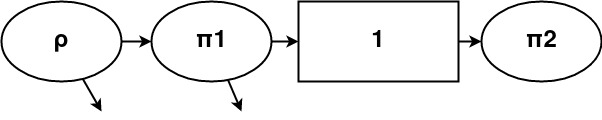
(заочная форма обучения)

Преподаватель Мельник Н.И.

Минск 2020

1. Исходные данные

P-схема



|  |  |  |  |
| --- | --- | --- | --- |
| ρ | π1 | π2 | Цель исследования |
| 0.7 | 0.7 | 0.75 | А , W с , L с |

1. Анализ задания

Элементы:

– источник с вероятностью просеивания и дисциплиной отбрасывания заявки

– канал обработки заявок с вероятностью просеивания π1 и дисциплиной отбрасывания заявки

1

– очередь на 1 заявку

– канал обработки заявок с вероятностью просеивания π2

Граф состояний кодируется трехкомпонентным вектором P1 N P2

Кодирование состояний:

N – количество заявок, находящихся в накопителе {0,1}:

0 – заявок в очереди на обслуживание нет

1 – одна заявка в очереди;

P1 и P2 – состояние каналов обслуживания {0,1}:

0 – канал свободен;

1 – канал занят обслуживанием заявки.

1. Найдём состояния, в которых может быть система:

000, 001, **010**, 100, 101, 011, **110**, 111.

Состояния, выделенные жирным шрифтом невозможны так как при пустом обработчике после, в накопителе не может оставаться функция.

1. Построим матрицу переходов в виде таблицы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| zk | zk | | | | |
| z1 | z2 | … | zk-1 | zk |
| z1 | р1,1 | р1,2 | … | р1,k-1 | р1,k |
| z2 | р2,1 | р2,2 | … | р2,k-1 | р2,k |
| … | … | … | … |  | … |
| zk | рk,1 | рk,2 | … | рk,k-1 | рk,k |

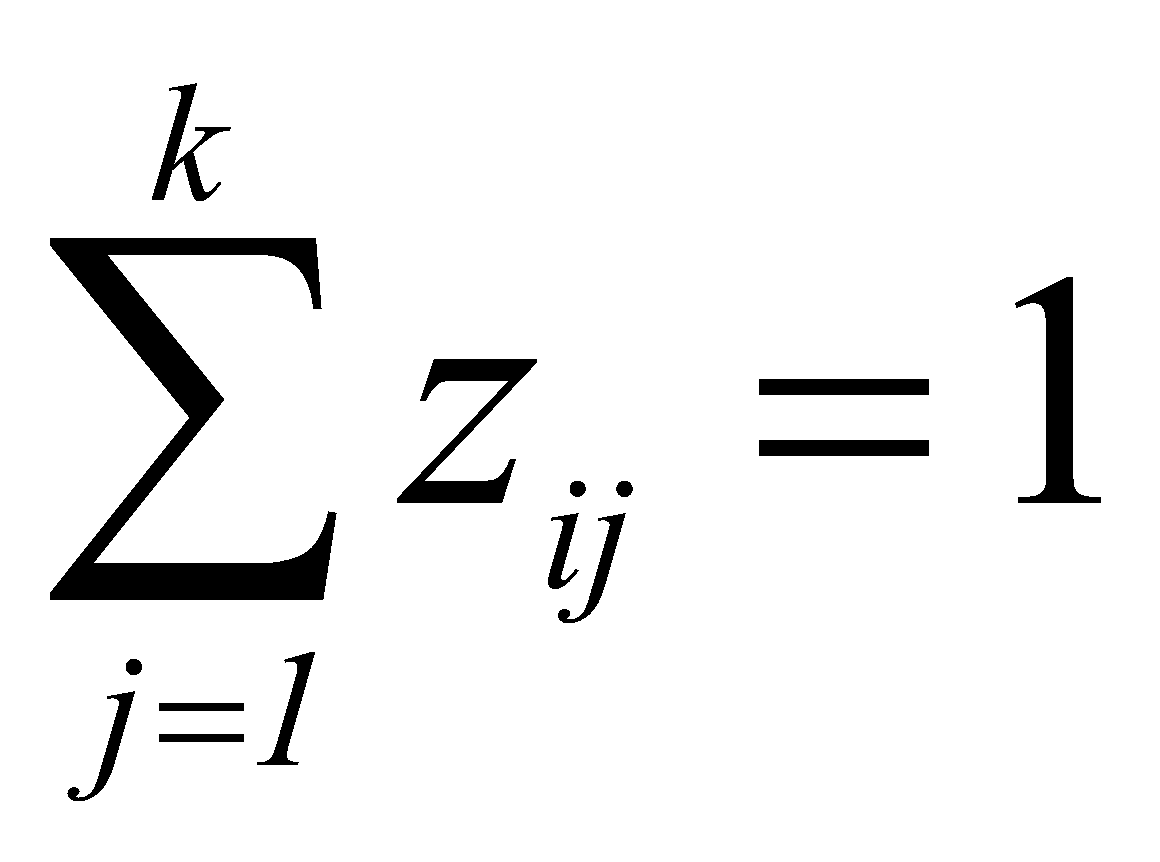
Где 

Таблица 1 – Матрица возможности переходов

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | 1 | 2 | 3 | 4 | 5 | 6 |
| 000 | 001 | 100 | 101 | 011 | 111 |
| 1 | 000 | p1,1 | 0 | p1,3 | 0 | 0 | 0 |
| 2 | 001 | p2,1 | p2,2 | p2,3 | p2,4 | 0 | 0 |
| 3 | 100 | 0 | p3,2 | p3,3 | p3,4 | 0 | 0 |
| 4 | 101 | 0 | p4,2 | p4,3 | p4,4 | p4,5 | p4,6 |
| 5 | 011 | 0 | p5,2 | 0 | p5,4 | p5,5 | p5,6 |
| 6 | 111 | 0 | p6,2 | 0 | p6,4 | p6,6 | p6,6 |

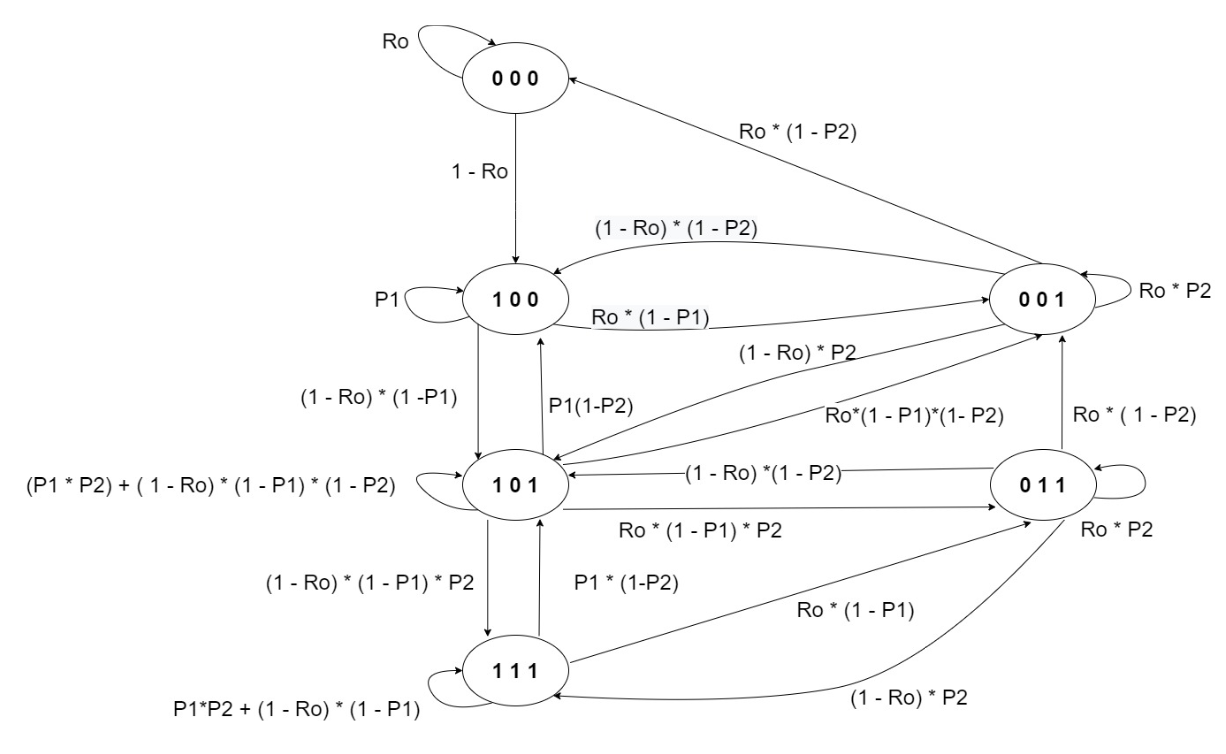
1. Определим вероятности переходов и заполним таблицу

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | 1 | 2 | 3 | 4 | 5 | 6 |
| 000 | 001 | 100 | 101 | 011 | 111 |
| 1 | 000 | ρ | 0 | 1 - ρ | 0 | 0 | 0 |
| 2 | 001 | ρ \* (1 - π2) | ρ \* π2 | (1 -ρ) \* (1 - π2) | (1 -ρ) \* π2 | 0 | 0 |
| 3 | 100 | 0 | ρ \*(1- π1) | π1 | (1 -ρ) \* (1 – π1) | 0 | 0 |
| 4 | 101 | 0 | ρ \*(1 -π1) \* (1 - π2) | π1 \* (1 - π2) | π1 \* π2+(1 – ρ) \* (1 – p1) \* (1 – p2) | ρ \*(1- π1) \* π2 | (1 -ρ) \* (1 – π1)\* π2 |
| 5 | 011 | 0 | ρ \* (1 - π2) | 0 | (1 - ρ) \* (1 - π2) | ρ \* π2 | (1 – ρ) \* π2 |
| 6 | 111 | 0 | 0 | 0 | π1 \* (1 - π2) | ρ \*(1 - π1) | π1 \* π2+(1 – ρ) \* (1 – p1) |

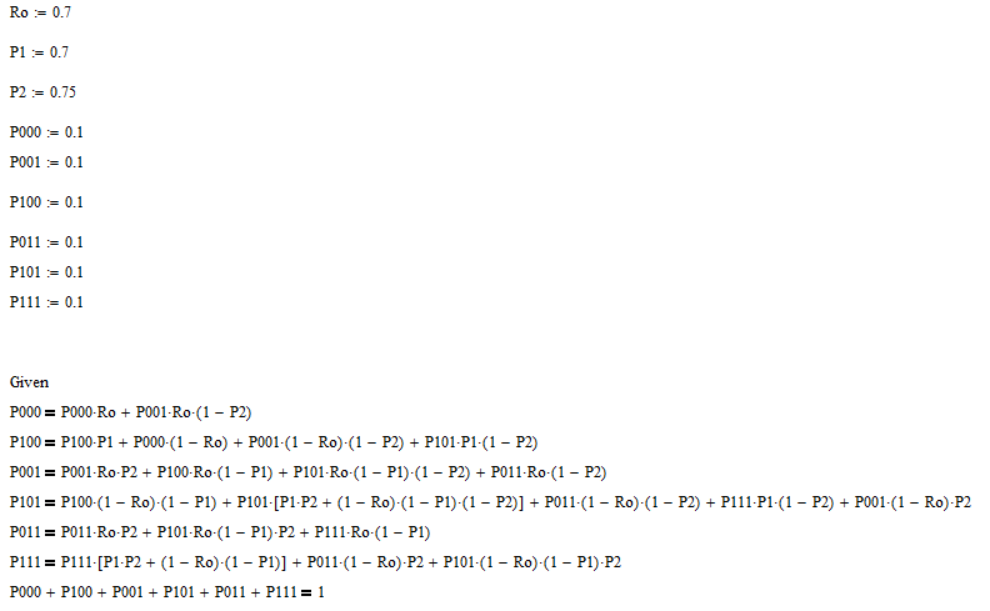
1. Подставим значения

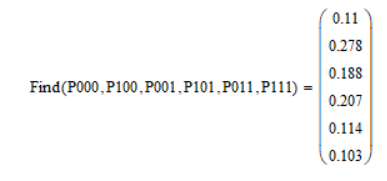
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | 1 | 2 | 3 | 4 | 5 | 6 |
| 000 | 001 | 100 | 101 | 011 | 111 |
| 1 | 000 | 0.7 | 0 | 0.3 | 0 | 0 | 0 |
| 2 | 001 | 0.175 | 0.525 | 0.075 | 0.225 | 0 | 0 |
| 3 | 100 | 0 | 0.21 | 0.7 | 0.9 | 0 | 0 |
| 4 | 101 | 0 | 0.053 | 0.175 | 0.547 | 0.158 | 0.067 |
| 5 | 011 | 0 | 0.175 | 0 | 0.075 | 0.525 | 0.225 |
| 6 | 111 | 0 | 0 | 0 | 0.175 | 0.21 | 0.615 |

1. Построим граф состояний



1. Построение аналитической модели
2. Решение системы в MathCad:





Абсолютная пропускная способность:

A = (1 - π2) \* (1 – P100 – P000) = 0.153

Среднее число заявок, находящихся в системе:

Lc = 0\*P000 + 1\*(P100 + P001) + 2\*(P101 + P011) + 3\*P111 = 1.419

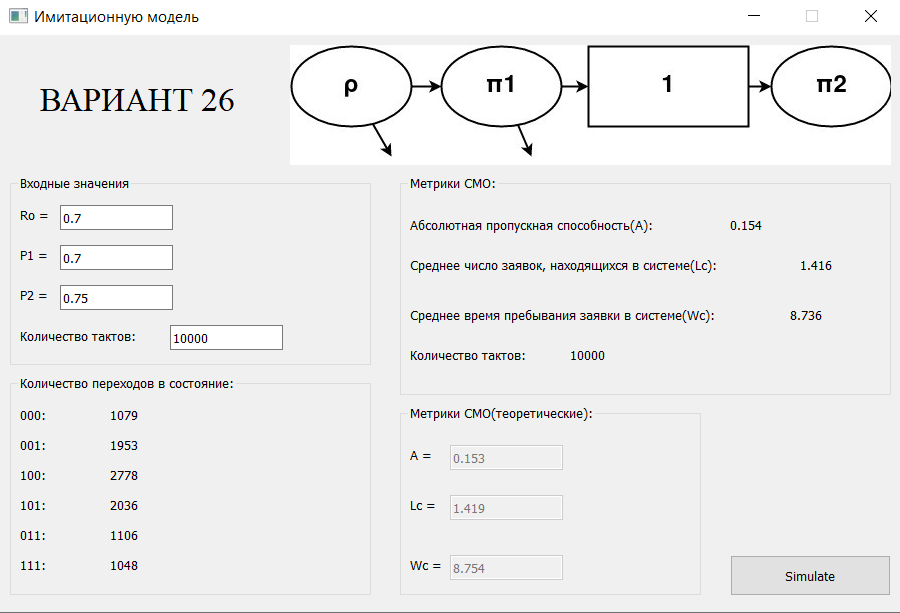
Среднее время пребывания заявки в системе:

Wс = (1 / (1 – π1)) + (P001 + P101 + 2\*(P111 + P011)) / A = 8.754

1. Исследование имитационной модели

Для СМО из задания построить имитационную модель и исследовать ее (разработать алгоритм и написать имитирующую программу, предусматривающую сбор и статистическую обработку данных для получения оценок заданных характеристик СМО). Распределение интервалов времени между заявками во входном потоке и интервалов времени обслуживания – геометрическое с соответствующим параметром (p, P1, P2).

1. Результат работы программы:



1. Вывод по текущему исследованию:

В ходе лабораторной работы была аналитически смоделирована дискретно- стохастическая СМО и разработана программа, имитирующая поведение данной СМО. Построенная модель позволяет статистически подсчитать характеристики СМО. Статистическое значение искомой характеристики оказывается близким к теоретически рассчитанному. Значит имитационная модель построена верно. Было также замечено, что на выходные данные влияют параметры СМО, такие как**p**, **P1**, **P2**.

1. Листинг программы:

*###main.py – точка входа в программу*

*import* sys

*from* PyQt5 *import* QtWidgets

*import* design

*from* smo *import* \*

class ExampleApp(QtWidgets.QMainWindow, design.Ui\_Dialog):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.setupUi(self)

        self.SimulateBtn.clicked.connect(self.Calculate)

    def Calculate(self):

        labels\_counter = [

        self.label000counter,

        self.label001counter,

        self.label100counter,

        self.label101counter,

        self.label011counter,

        self.label111counter,

        ]

        Ro = self.RoLineEdit

        P1 = self.P1LineEdit

        P2 = self.P2LineEdit

        Ticks = self.TicksLineEdit

        Acounter = self.Acounter

        Lccounter = self.Lccounter

        Wccounter = self.Wccounter

        Tickscounter = self.Tickscounter

        main\_event(float(Ro.text()), float(P1.text()), float(P2.text()), int(Ticks.text()), labels\_counter, Acounter, Lccounter, Wccounter, Tickscounter)

def main():

    app = QtWidgets.QApplication(sys.argv)

    window = ExampleApp()

    window.show()

    app.exec\_()

*if* \_\_name\_\_ == '\_\_main\_\_':

    main()

*###smo.py - логическая основа программы*

*import* random

def change\_state(A, state, states\_counter, states):

    state = random.choices(list(states.keys()), weights=A[states[state]])[0]

    states\_counter[state] = states\_counter[state] + 1

*return* state, states\_counter

def make\_cacl(tick, Acounter, Lccounter, Wccounter, Tickscounter, P1, P2, states\_counter):

    P000 = states\_counter['000']

    P100 = states\_counter['100']

    P001 = states\_counter['001']

    P101 = states\_counter['101']

    P111 = states\_counter['111']

    P011 = states\_counter['011']

    P000 = P000  / tick *if* P000 *else* 0

    P100 = P100  / tick *if* P100 *else* 0

    P001 = P001  / tick *if* P001 *else* 0

    P101 = P101  / tick *if* P101 *else* 0

    P111 = P111  / tick *if* P111 *else* 0

    P011 = P011  / tick *if* P011 *else* 0

    Ac = (1 - P2) \* (1 - P000 - P100)

    Lc = sum([k.count('1') \* v  *for* k,v in states\_counter.items()]) / tick

    Wc = (1 / (1 - P1)) + (P001 + P101 + 2\*(P111 + P011)) / Ac

    Acounter.setText(str(round(Ac, 3)))

    Lccounter.setText(str(round(Lc, 3)))

    Wccounter.setText(str(round(Wc, 3)))

    Tickscounter.setText(str(round(tick, 3)))

def main\_event(Ro, P1, P2, Ticks, labels\_counter, Acounter, Lccounter, Wccounter, Tickscounter):

    A = [

    [Ro, 0,  1-Ro,  0, 0, 0],

    [Ro\*(1 - P2), Ro\*P2,  (1 - Ro) \* ( 1- P2), (1 - Ro)\*P2,  0, 0],

    [0, Ro\*(1 - P1),  P1,  (1 - Ro)\*(1 - P1), 0, 0],

    [0, Ro\* (1-P1)\*(1-P2),  P1\*(1-P2), P1\*P2 + (1 - Ro)\*(1 - P1)\*(1 - P2),  Ro \*(1-P1)\*P2, (1 - Ro) \* (1 - P1)],

    [0, Ro\*(1-P2),  0,  (1-Ro)\*(1-P2), Ro\*P2, (1-Ro)\*P2],

    [0, 0, 0, P1\*(1-P2), Ro \* (1-P1), P1\*P2 + (1 - Ro) \* (1 - P1)]

    ]

    state = '000'

    states = {'000': 0, '001': 1, '100': 2, '101': 3, '011': 4, '111': 5}

    states\_counter = {'000': 0, '001': 0, '100': 0, '101': 0, '011': 0, '111': 0}

*for* tick in range(1, Ticks + 1):

*if* not P2:

*for* i in A:

                i[states['011']] = 0

                i[states['111']] = 0

        state, states\_counter = change\_state(A, state, states\_counter, states)

*for* label\_counter, state\_counter in zip(labels\_counter, list(states\_counter.values())):

        label\_counter.setText(str(state\_counter))

    make\_cacl(Ticks, Acounter, Lccounter, Wccounter, Tickscounter, P1, P2, states\_counter)

*###design.py - сгенерированный графический интерфейс*

*# -\*- coding: utf-8 -\*-*

*# Form implementation generated from reading ui file 'SMO.ui'*

*#*

*# Created by: PyQt5 UI code generator 5.11.3*

*#*

*# WARNING! All changes made in this file will be lost!*

*from* PyQt5 *import* QtCore, QtGui, QtWidgets

class Ui\_Dialog(object):

    def setupUi(self, Dialog):

        Dialog.setObjectName("Dialog")

        Dialog.setEnabled(True)

        Dialog.resize(901, 577)

        self.setFixedSize(901,577)

        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed, QtWidgets.QSizePolicy.Fixed)

        sizePolicy.setHorizontalStretch(0)

        sizePolicy.setVerticalStretch(0)

        sizePolicy.setHeightForWidth(Dialog.sizePolicy().hasHeightForWidth())

        Dialog.setSizePolicy(sizePolicy)

        Dialog.setContextMenuPolicy(QtCore.Qt.ActionsContextMenu)

        self.SimulateBtn = QtWidgets.QPushButton(Dialog)

        self.SimulateBtn.setGeometry(QtCore.QRect(730, 520, 161, 41))

        self.SimulateBtn.setObjectName("SimulateBtn")

        self.PicLabel = QtWidgets.QLabel(Dialog)

        self.PicLabel.setGeometry(QtCore.QRect(290, 0, 601, 141))

        self.PicLabel.setText("")

        self.PicLabel.setPixmap(QtGui.QPixmap("task.jpg"))

        self.PicLabel.setObjectName("PicLabel")

        self.VarLabel = QtWidgets.QLabel(Dialog)

        self.VarLabel.setGeometry(QtCore.QRect(40, 40, 201, 51))

        font = QtGui.QFont()

        font.setFamily("Times New Roman")

        font.setPointSize(20)

        self.VarLabel.setFont(font)

        self.VarLabel.setObjectName("VarLabel")

        self.InputBox = QtWidgets.QGroupBox(Dialog)

        self.InputBox.setGeometry(QtCore.QRect(10, 140, 361, 191))

        self.InputBox.setObjectName("InputBox")

        self.TicksLineEdit = QtWidgets.QLineEdit(self.InputBox)

        self.TicksLineEdit.setValidator(QtGui.QIntValidator(1, 999999))

        self.TicksLineEdit.setGeometry(QtCore.QRect(160, 150, 113, 25))

        self.TicksLineEdit.setObjectName("TicksLineEdit")

        self.RoLabel = QtWidgets.QLabel(self.InputBox)

        self.RoLabel.setEnabled(True)

        self.RoLabel.setGeometry(QtCore.QRect(10, 30, 41, 19))

        self.RoLabel.setObjectName("RoLabel")

        self.RoLineEdit = QtWidgets.QLineEdit(self.InputBox)

        self.RoLineEdit.setValidator(QtGui.QRegExpValidator(QtCore.QRegExp('^(0(\.\d{1,2}))')))

        self.RoLineEdit.setGeometry(QtCore.QRect(50, 30, 113, 25))

        self.RoLineEdit.setObjectName("RoLineEdit")

        self.P1Label = QtWidgets.QLabel(self.InputBox)

        self.P1Label.setGeometry(QtCore.QRect(10, 70, 41, 19))

        self.P1Label.setObjectName("P1Label")

        self.TiksLabel = QtWidgets.QLabel(self.InputBox)

        self.TiksLabel.setGeometry(QtCore.QRect(10, 150, 151, 21))

        self.TiksLabel.setObjectName("TiksLabel")

        self.P2Label = QtWidgets.QLabel(self.InputBox)

        self.P2Label.setGeometry(QtCore.QRect(10, 110, 41, 19))

        self.P2Label.setObjectName("P2Label")

        self.P1LineEdit = QtWidgets.QLineEdit(self.InputBox)

        self.P1LineEdit.setValidator(QtGui.QRegExpValidator(QtCore.QRegExp('^(0(\.\d{1,2}))')))

        self.P1LineEdit.setGeometry(QtCore.QRect(50, 70, 113, 25))

        self.P1LineEdit.setObjectName("P1LineEdit")

        self.P2LineEdit = QtWidgets.QLineEdit(self.InputBox)

        self.P2LineEdit.setValidator(QtGui.QRegExpValidator(QtCore.QRegExp('^(0(\.\d{1,2}))')))

        self.P2LineEdit.setGeometry(QtCore.QRect(50, 110, 113, 25))

        self.P2LineEdit.setObjectName("P2LineEdit")

        self.StateBox = QtWidgets.QGroupBox(Dialog)

        self.StateBox.setGeometry(QtCore.QRect(10, 340, 361, 221))

        self.StateBox.setObjectName("StateBox")

        self.label000 = QtWidgets.QLabel(self.StateBox)

        self.label000.setGeometry(QtCore.QRect(10, 30, 68, 19))

        self.label000.setObjectName("label000")

        self.label001 = QtWidgets.QLabel(self.StateBox)

        self.label001.setGeometry(QtCore.QRect(10, 60, 68, 19))

        self.label001.setObjectName("label001")

        self.label100 = QtWidgets.QLabel(self.StateBox)

        self.label100.setGeometry(QtCore.QRect(10, 90, 68, 19))

        self.label100.setObjectName("label100")

        self.label101 = QtWidgets.QLabel(self.StateBox)

        self.label101.setGeometry(QtCore.QRect(10, 120, 68, 19))

        self.label101.setObjectName("label101")

        self.label011 = QtWidgets.QLabel(self.StateBox)

        self.label011.setGeometry(QtCore.QRect(10, 150, 68, 19))

        self.label011.setObjectName("label011")

        self.label111 = QtWidgets.QLabel(self.StateBox)

        self.label111.setGeometry(QtCore.QRect(10, 180, 68, 19))

        self.label111.setObjectName("label111")

        self.label000counter = QtWidgets.QLabel(self.StateBox)

        self.label000counter.setGeometry(QtCore.QRect(100, 30, 68, 19))

        self.label000counter.setObjectName("label000counter")

        self.label001counter = QtWidgets.QLabel(self.StateBox)

        self.label001counter.setGeometry(QtCore.QRect(100, 60, 68, 19))

        self.label001counter.setObjectName("label001counter")

        self.label100counter = QtWidgets.QLabel(self.StateBox)

        self.label100counter.setGeometry(QtCore.QRect(100, 90, 68, 19))

        self.label100counter.setObjectName("label100counter")

        self.label101counter = QtWidgets.QLabel(self.StateBox)

        self.label101counter.setGeometry(QtCore.QRect(100, 120, 68, 19))

        self.label101counter.setObjectName("label101counter")

        self.label011counter = QtWidgets.QLabel(self.StateBox)

        self.label011counter.setGeometry(QtCore.QRect(100, 150, 68, 19))

        self.label011counter.setObjectName("label011counter")

        self.label111counter = QtWidgets.QLabel(self.StateBox)

        self.label111counter.setGeometry(QtCore.QRect(100, 180, 68, 19))

        self.label111counter.setObjectName("label111counter")

        self.MetricsBox = QtWidgets.QGroupBox(Dialog)

        self.MetricsBox.setGeometry(QtCore.QRect(400, 140, 491, 221))

        self.MetricsBox.setObjectName("MetricsBox")

        self.ALabel = QtWidgets.QLabel(self.MetricsBox)

        self.ALabel.setEnabled(True)

        self.ALabel.setGeometry(QtCore.QRect(10, 40, 311, 19))

        self.ALabel.setObjectName("ALabel")

        self.LcLabel = QtWidgets.QLabel(self.MetricsBox)

        self.LcLabel.setGeometry(QtCore.QRect(10, 80, 381, 19))

        self.LcLabel.setObjectName("LcLabel")

        self.WcLabel = QtWidgets.QLabel(self.MetricsBox)

        self.WcLabel.setGeometry(QtCore.QRect(10, 130, 371, 19))

        self.WcLabel.setObjectName("WcLabel")

        self.TicksLabel = QtWidgets.QLabel(self.MetricsBox)

        self.TicksLabel.setGeometry(QtCore.QRect(10, 170, 151, 19))

        self.TicksLabel.setObjectName("TicksLabel")

        self.Acounter = QtWidgets.QLabel(self.MetricsBox)

        self.Acounter.setGeometry(QtCore.QRect(330, 40, 68, 19))

        self.Acounter.setObjectName("Acounter")

        self.Lccounter = QtWidgets.QLabel(self.MetricsBox)

        self.Lccounter.setGeometry(QtCore.QRect(400, 80, 61, 20))

        self.Lccounter.setObjectName("Lccounter")

        self.Wccounter = QtWidgets.QLabel(self.MetricsBox)

        self.Wccounter.setGeometry(QtCore.QRect(390, 130, 68, 19))

        self.Wccounter.setObjectName("Wccounter")

        self.Tickscounter = QtWidgets.QLabel(self.MetricsBox)

        self.Tickscounter.setGeometry(QtCore.QRect(170, 170, 68, 19))

        self.Tickscounter.setObjectName("Tickscounter")

        self.DefaultMetricBox = QtWidgets.QGroupBox(Dialog)

        self.DefaultMetricBox.setGeometry(QtCore.QRect(400, 370, 301, 191))

        self.DefaultMetricBox.setObjectName("DefaultMetricBox")

        self.DefAlabel = QtWidgets.QLabel(self.DefaultMetricBox)

        self.DefAlabel.setGeometry(QtCore.QRect(10, 40, 31, 19))

        self.DefAlabel.setObjectName("DefAlabel")

        self.DefLcLabel = QtWidgets.QLabel(self.DefaultMetricBox)

        self.DefLcLabel.setGeometry(QtCore.QRect(10, 90, 31, 19))

        self.DefLcLabel.setObjectName("DefLcLabel")

        self.DefWcLAbel = QtWidgets.QLabel(self.DefaultMetricBox)

        self.DefWcLAbel.setGeometry(QtCore.QRect(10, 150, 51, 19))

        self.DefWcLAbel.setObjectName("DefWcLAbel")

        self.DefALineEdit = QtWidgets.QLineEdit(self.DefaultMetricBox)

        self.DefALineEdit.setEnabled(False)

        self.DefALineEdit.setGeometry(QtCore.QRect(50, 40, 113, 25))

        self.DefALineEdit.setReadOnly(True)

        self.DefALineEdit.setObjectName("DefALineEdit")

        self.DefLcLineEdit = QtWidgets.QLineEdit(self.DefaultMetricBox)

        self.DefLcLineEdit.setEnabled(False)

        self.DefLcLineEdit.setGeometry(QtCore.QRect(50, 90, 113, 25))

        self.DefLcLineEdit.setObjectName("DefLcLineEdit")

        self.DefWcLineEdit = QtWidgets.QLineEdit(self.DefaultMetricBox)

        self.DefWcLineEdit.setEnabled(False)

        self.DefWcLineEdit.setGeometry(QtCore.QRect(50, 150, 113, 25))

        self.DefWcLineEdit.setObjectName("DefWcLineEdit")

        self.InputBox.raise\_()

        self.SimulateBtn.raise\_()

        self.PicLabel.raise\_()

        self.VarLabel.raise\_()

        self.StateBox.raise\_()

        self.MetricsBox.raise\_()

        self.DefaultMetricBox.raise\_()

        self.retranslateUi(Dialog)

        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):

        \_translate = QtCore.QCoreApplication.translate

        Dialog.setWindowTitle(\_translate("Dialog", "Имитационную модель"))

        self.SimulateBtn.setText(\_translate("Dialog", "Simulate"))

        self.VarLabel.setText(\_translate("Dialog", "ВАРИАНТ 26"))

        self.InputBox.setTitle(\_translate("Dialog", "Входные значения"))

        self.TicksLineEdit.setText(\_translate("Dialog", "10000"))

        self.RoLabel.setText(\_translate("Dialog", "Ro = "))

        self.RoLineEdit.setText(\_translate("Dialog", "0.7"))

        self.P1Label.setText(\_translate("Dialog", "P1 ="))

        self.TiksLabel.setText(\_translate("Dialog", "Количество тактов:"))

        self.P2Label.setText(\_translate("Dialog", "P2 = "))

        self.P1LineEdit.setText(\_translate("Dialog", "0.7"))

        self.P2LineEdit.setText(\_translate("Dialog", "0.75"))

        self.StateBox.setTitle(\_translate("Dialog", "Количество переходов в состояние:"))

        self.label000.setText(\_translate("Dialog", "000:"))

        self.label001.setText(\_translate("Dialog", "001:"))

        self.label100.setText(\_translate("Dialog", "100:"))

        self.label101.setText(\_translate("Dialog", "101:"))

        self.label011.setText(\_translate("Dialog", "011:"))

        self.label111.setText(\_translate("Dialog", "111:"))

        self.label000counter.setText(\_translate("Dialog", "0"))

        self.label001counter.setText(\_translate("Dialog", "0"))

        self.label100counter.setText(\_translate("Dialog", "0"))

        self.label101counter.setText(\_translate("Dialog", "0"))

        self.label011counter.setText(\_translate("Dialog", "0"))

        self.label111counter.setText(\_translate("Dialog", "0"))

        self.MetricsBox.setTitle(\_translate("Dialog", "Метрики СМО:"))

        self.ALabel.setText(\_translate("Dialog", "Абсолютная пропускная способность(A):"))

        self.LcLabel.setText(\_translate("Dialog", "Среднее число заявок, находящихся в системе(Lc):"))

        self.WcLabel.setText(\_translate("Dialog", "Среднее время пребывания заявки в системе(Wc):"))

        self.TicksLabel.setText(\_translate("Dialog", "Количество тактов:"))

        self.Acounter.setText(\_translate("Dialog", "0"))

        self.Lccounter.setText(\_translate("Dialog", "0"))

        self.Wccounter.setText(\_translate("Dialog", "0"))

        self.Tickscounter.setText(\_translate("Dialog", "0"))

        self.DefaultMetricBox.setTitle(\_translate("Dialog", "Метрики СМО(теоретические):"))

        self.DefAlabel.setText(\_translate("Dialog", "A = "))

        self.DefLcLabel.setText(\_translate("Dialog", "Lc ="))

        self.DefWcLAbel.setText(\_translate("Dialog", "Wc = "))

        self.DefALineEdit.setText(\_translate("Dialog", "0.153"))

        self.DefLcLineEdit.setText(\_translate("Dialog", "1.419"))

        self.DefWcLineEdit.setText(\_translate("Dialog", "8.754"))