МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Пятигорский институт (филиал) СКФУ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ «МОДЕЛИ И МЕТОДЫ ИССЛЕДОВАНИЯ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ И СИСТЕМ»

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Направление подготовки 09.04.02 «Информационные системы и технологии»
Направленность (профиль) «Технологии работы с данными и знаниями, анализ информации»

Пятигорск, 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1. Цель и задачи освоения дисциплины (модуля)

Целью освоения дисциплины «Модели и методы исследования информационных процессов и систем» является получение знаний, умений и навыков по использованию методов исследования и моделирования информационных процессов и систем при проведении научных исследований и решении практических задач, а также формирование на их основе компетенций будущего магистра по направлению подготовки 09.04.02. – Информационные системы и технологии и направленности (профилю) «Технологии работы с данными и знаниями, анализ информации». Освоение методологии и технологии моделирования, в первую очередь компьютерного, информационных процессов в различных системах. Приобретение навыков профессионального использования современных пакетов имитационного моделирования при разработке моделей задач из различных сфер деятельности.

Задачи дисциплины: использование методологии системного анализа для исследования информационных процессов, систем и технологий; моделирование информационных процессов и систем; изучение современных подходов к формализации информационных систем; использование информационных технологий при анализе и синтезе информационных систем и инструментальных средств моделирования информационных процессов и систем

2. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
ОПК-7 Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза	ОПК-7.1 Применяет методы научных исследований и математического моделирования при решении задач анализа и синтеза	Правильно использует: методы анализа и синтеза информационных систем; формальные модели систем; средства структурного анализа; методологию структурного системного анализа и проектирования. Грамотно применяет методы научных исследований и математического моделирования при решении задач анализа и синтеза

распределенных информационных систем и систем поддержки принятия решений;	распределенных информационных систем и систем поддержки принятия решений	распределенных информационных систем и систем поддержки принятия решений Выбирает адекватные методы научного поиска и интеллектуального анализа научной информации при решении новых задач, методы анализа и синтеза информационных систем, распределенных информационных систем и систем поддержки принятия решений. Адекватно использует математический аппарат для решения задач в области информационных систем и технологий
	ОПК-7.2 Разрабатывает математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений	Анализирует динамические оптимизационные модели, модели предметных областей информационных систем, математические модели информационных процессов Разрабатывает модели процессов и объектов информационной среды, аналитические и имитационные модели предметных областей. Выбирает адекватные модели и средства разработки архитектуры информационных систем. Использует адекватные основные приемы по исследованию информационных систем и технологий
ОПК-8 Способен осуществлять эффективное управление разработкой программных средств и проектов	ОПК-8.1: Осуществляет управление работами по выявлению и анализу требований к программным средствам и проектам	Анализирует: принципы выявления, разработки, документирования требований в ИТ проектах; принципы изменения и планирования требований к программным средствам и проектам. Выполняет: обзор прикладных и информационных процессов; анализ прикладных и информационных процессов; реинжиниринг

		прикладных и информационных процессов. Демонстрирует практические навыки: выбора необходимого программного обеспечения; планирования и разработки программного обеспечения
	ОПК-8.2: Проводит мониторинг и управляет работами проекта в ИТ области	Грамотно использует: понятия жизненного цикла проекта: инициацию, планирование, исполнение, мониторинг и контроль, закрытие; основную стратегию разработки программных средств и проектов, особенности процессного подхода к управлению прикладными ИС и современные ИКТ в процессном управлении Осуществляет: эффективное управление проектами ИС на всех стадиях жизненного цикла, оценивание эффективности и качества проекта; применение современных методов управления проектами и сервисами ИС; обеспечение эффективного контроля и регулирования работ, а также управление изменениями Демонстрирует практические навыки по анализу спецификации, мониторингу, управлению и контролю работами проекта в ИТ области

ЛАБОРАТОРНАЯ РАБОТА 1
Моделирование процессов и систем в пакете Simulink системы MATLAB

Цель и содержание: изучить основные приемы работы с системой компьютерной математики MATLAB приобрести навыки построения моделей систем в пакете Simulink

Организационная форма занятий: решение проблемных задач, разбор конкретных ситуаций

Вопросы для обсуждения на лабораторном занятии: построение простейших моделей в пакете Simulink системы MATLAB.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Пакет прикладных программ MATLAB относится к инструментальным средствам, которые предназначены для решения задач в различных областях человеческой деятельности (научных, экономических, инженерных, физических и т. д.), позволяющим производить моделирование, разработку и отладку различных систем и устройств. Его используют более миллиона инженерных и научных работников, он работает на большинстве современных операционных систем, включая Linux, macOS, и Windows.

MATLAB имеет большое количество функций для анализа данных, практически из всех областей математики. MATLAB как язык программирования был разработан Кливом Моулером (Cleve Moler) в конце 1970-х годов. Язык MATLAB является высокоуровневым интерпретируемым языком программирования, включающим основанные на матрицах структуры данных, широкий спектр функций, интегрированную среду разработки, объектно-ориентированные возможности и интерфейсы к программам, написанным на других языках программирования.

Программы, написанные на MATLAB, бывают двух типов — функции и скрипты. Функции имеют входные и выходные аргументы, а также собственное рабочее пространство для хранения промежуточных результатов вычислений и переменных. Скрипты же используют общее рабочее пространство. Как скрипты, так и функции сохраняются в виде текстовых файлов и компилируются в машинный код динамически. Существует также возможность сохранять так называемые *pre-parsed* программы — функции и скрипты, обработанные в вид, удобный для машинного исполнения. В общем случае такие программы выполняются быстрее обычных, особенно если функция содержит команды построения графиков.

Основной особенностью языка MATLAB являются его широкие возможности по работе с матрицами

Интерфейс системы MATLAB представлен 1.1.

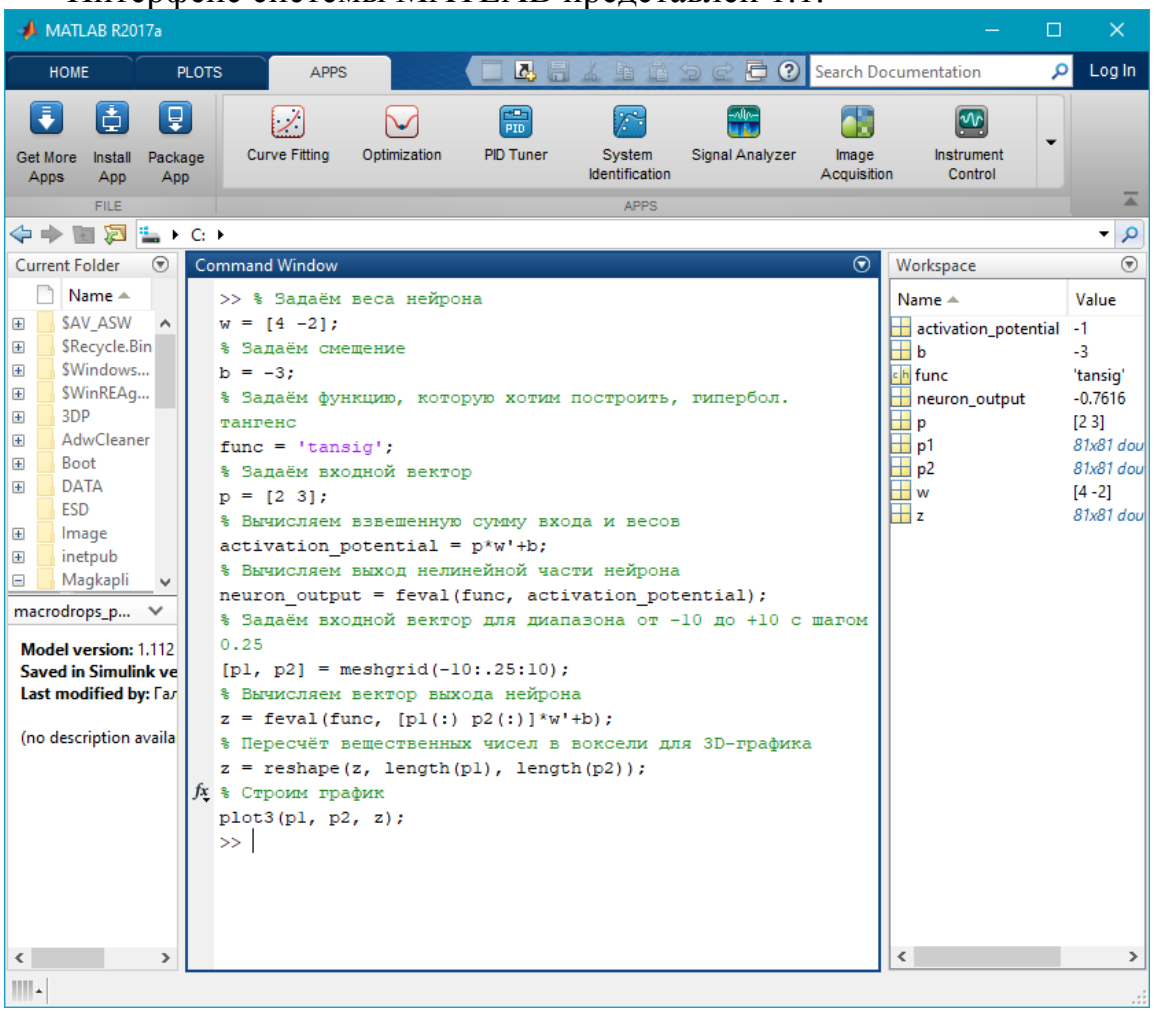


Рисунок 1.1 –Интерфейс системы MATLAB

Simulink – это подсистема имитационного математического моделирования динамических процессов. Simulink является составной частью пакета Matlab и полностью интегрирован с ним. Модели в Simulink состоят из набора графических блоков, представляющих компоненты объекта или какие-то функциональные элементы (источники сигналов различного вида, виртуальные регистрирующие приборы, средства анимации), и направленных связей между ними (линии распространения сигналов). Simulink позволяет осуществлять имитационное моделирование поведения различных динамических нелинейных систем.

Для начала работы необходимо создать новый файл Simulink model, в котором с помощью библиотек Simulink (рисунок 1.2, 1.3) и осуществить графическую сборку системы из отдельных блоков, хранящихся в библиотеках Simulink.

Рассмотрим некоторые блоки математических операций (рисунок 1.2):

- **Abs** предназначен для вычисления абсолютного значения входного сигнала;

- **Add** предназначен для вычисления алгебраической суммы двух (или более) сигналов;

Gain умножает входной сигнал на заданный коэффициент и т.д.

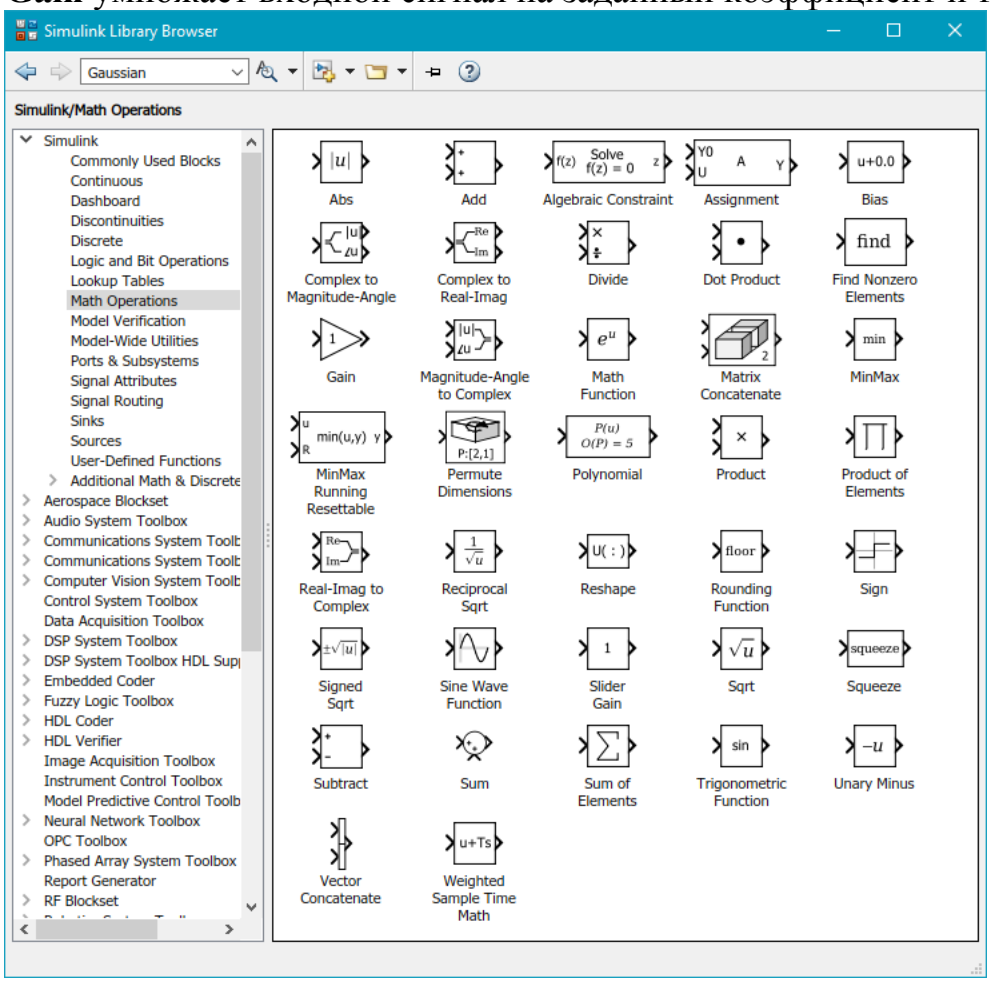


Рисунок 1.2 – Библиотеки Simulink. Блоки математических операций

Рассмотрим часто встречающиеся блоки (рисунок 1.3):

- **From File** – блок считывания данных из файла;
- **Sine Wave** формирует синусоидальный сигнал с заданной частотой, амплитудой, фазой и смещением;
- **Signal Generator** формирует один из четырех видов периодических сигналов (синусоидальный, прямоугольный, пилообразный, случайный сигналы);
- **Ramp** формирует линейный сигнал;
- **Step** формирует ступенчатый сигнал;
- **Random Number** формирует случайный сигнал с нормальным распределением уровня сигнала и т.д.

Аппаратура и материалы. Для выполнения лабораторной работы необходимо использовать следующее: аппаратное обеспечение: персональный компьютер и программное обеспечение: операционную систему Windows 10 и выше; Microsoft Office13 и выше, системы компьютерной математики Machcad и MATLAB R2017a и выше.

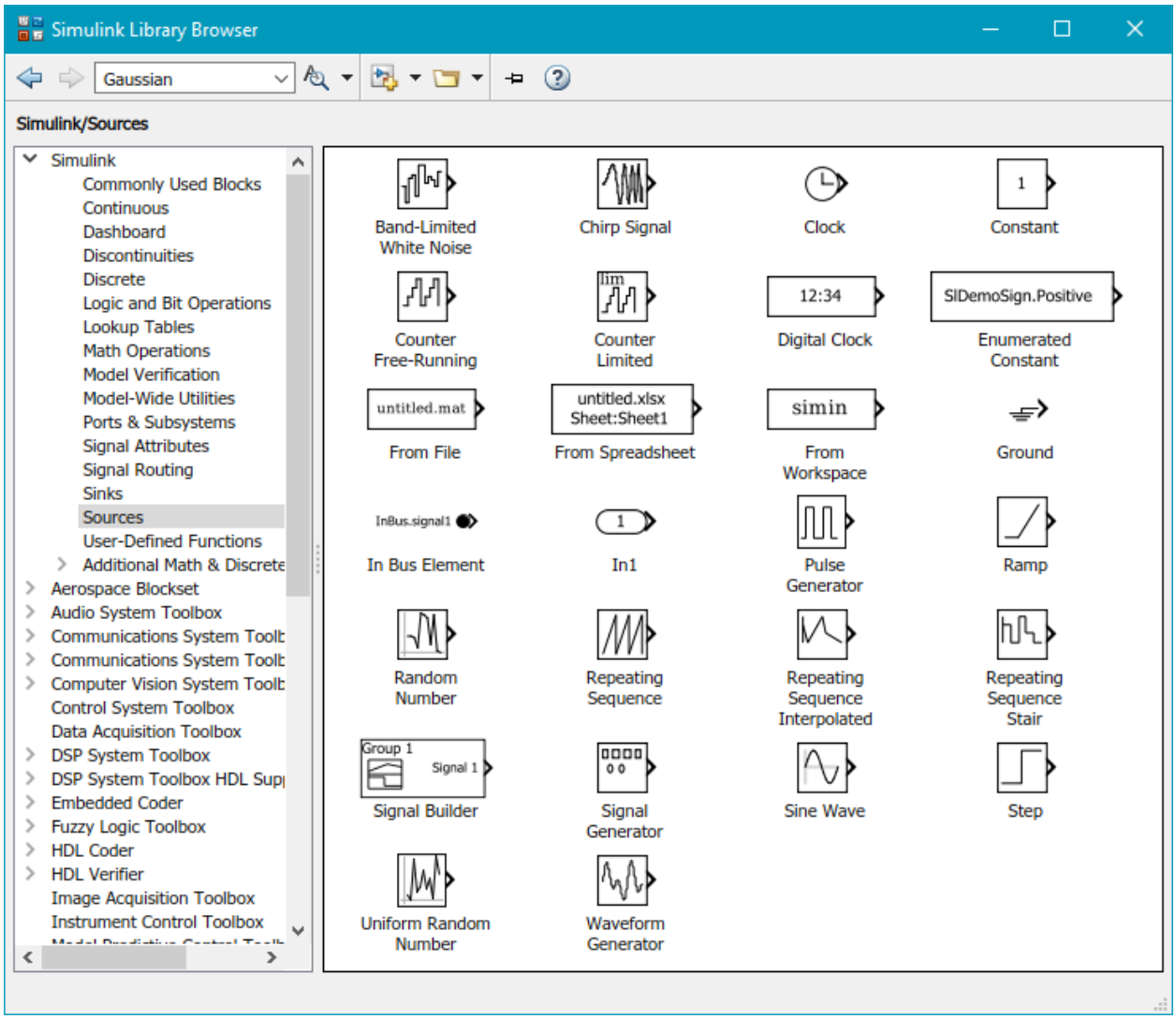


Рисунок 1.3 – Блоки библиотеки Simulink/Sources

Методика и порядок выполнения работы

Выполните предложенные задания, предварительно рассмотрев приведенные в примеры.

Задание 1.1. Построение простейших моделей

Построить модель сигнала вида $x(t) = A\sin(\omega t+\varphi)+t^m$ на интервале $[a; b]$, и результаты моделирования вывести на виртуальный осциллограф. Параметры сигналов приведены в таблице 1.1. Номер варианта соответствует номеру, под которым студент записан в списке группы.

На рисунке 1.4 приведен один из вариантов построения сигнала: $x(t)=1{,}5\sin(\pi t+\pi/3)+t^2$ с помощью функциональных блоков. Сигнал $1{,}5\sin(\pi t+\pi/3)$ задан в окне диалога параметров блока Sine Wave – Source Block Parameters: Sine Wave (рисунок 1.5).

Таблица 1.1 – Параметры сигнала

№ варианта	Амплитуда, A	Частота, ω	Фаза, φ	Степень, *m*	Интервал,

					$[a; b]$
1.	0,5	2π	$\pi/3$	2	[0; 3]
2.	2,5	$3\pi/4$	$\pi/6$	3	[0; 1]
3.	1,5	$2\pi/3$	$\pi/8$	2	[0; 5]
4.	4,5	$\pi/4$	$5\pi/4$	4	[0; 6]
5.	3,5	$3\pi/2$	$3\pi/5$	3	[0; 7]
6.	8,5	$\pi/8$	$3\pi/4$	1,5	[0; 4]
7.	7,5	$5\pi/4$	$2\pi/3$	3,5	[0; 1,5]
8.	9,5	$3\pi/5$	$\pi/4$	2	[0; 2,5]
9.	0,9	$\pi/3$	$3\pi/2$	4	[0; 1,8]
10.	0,45	$\pi/6$	$\pi/12$	3	[0; 3,2]

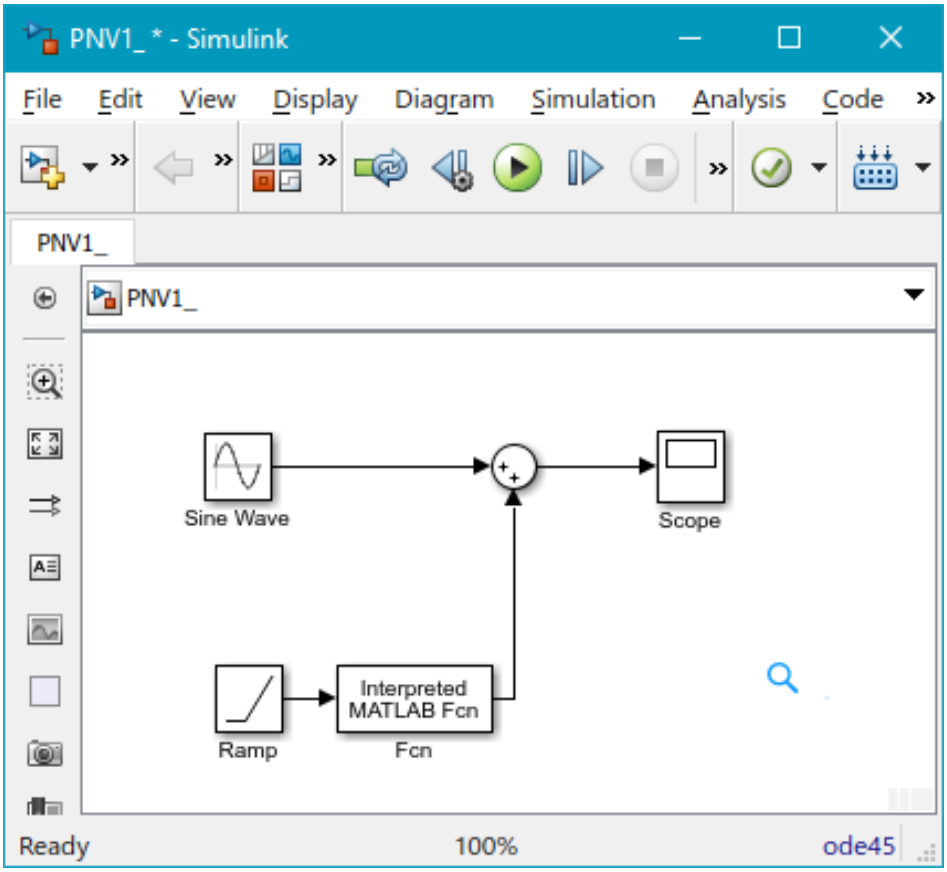


Рисунок 1.4 – Структурная схема модели сигнала $x(t)=1{,}5\sin(\pi t+\pi/3)+t^2$

Для сигнала t^2 использовались два блока – блок линейного сигнала и блок математической функции, для которого в соответствующем окне была введена функция в степени 2 (рисунок 1.6).

Интервал моделирования задан в пределах от 0 до 2 в диалоговом окне Configuration Parameters меню Simulation (рисунок 1.7).

Результаты моделирования работы выведены на экран виртуального осциллографа (рисунок 1.8).

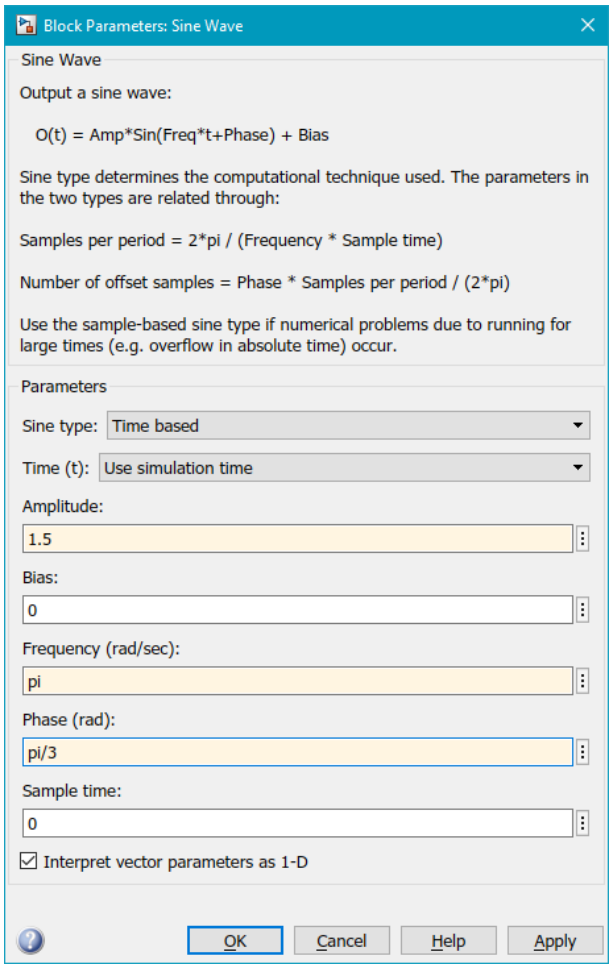


Рисунок 1.5– Окно диалога параметров блока Sine Wave с заданными параметрами сигнала

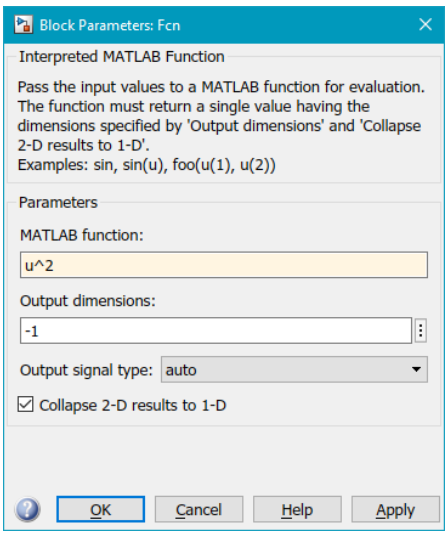


Рисунок 1.6 – Параметры функции t^2

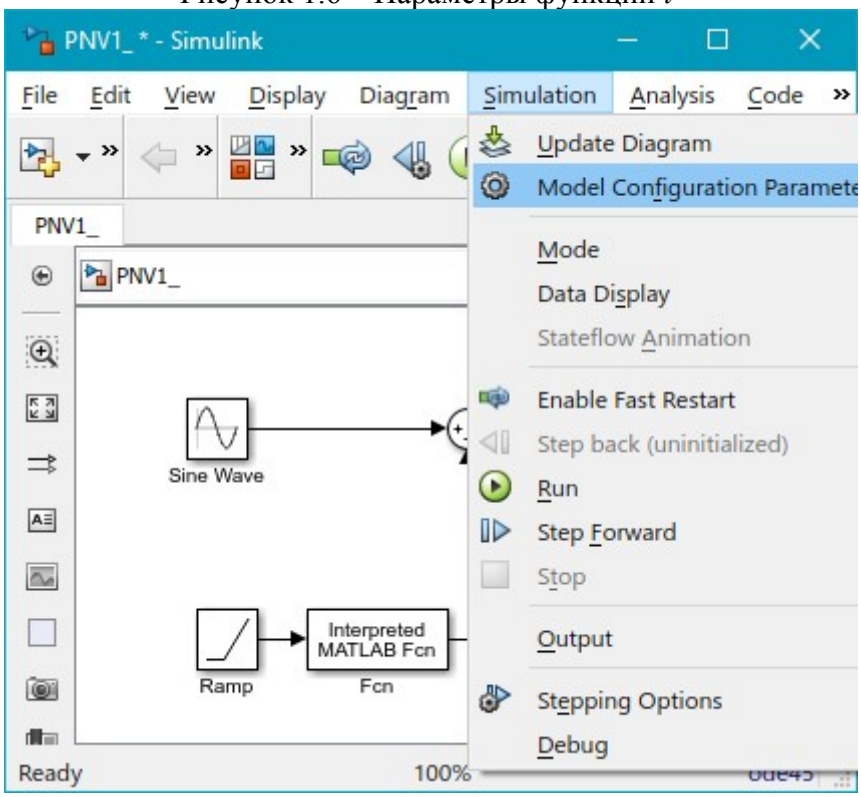


Рисунок 1.7 – Вызов окна диалога Configuration Parameters меню Simulation

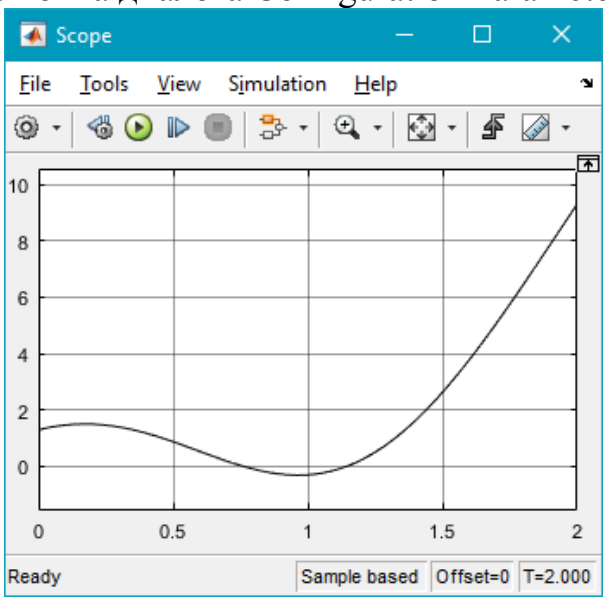


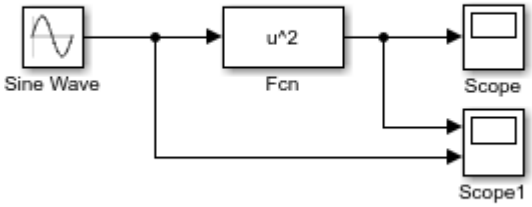
Рисунок 1.8 –График заданного сигнала на экране виртуального осциллографа

Задание 1.2. Моделирование сигнала заданной формы:

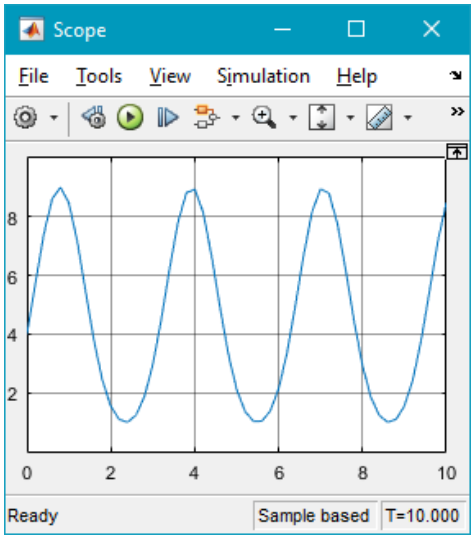
$x(t) = (2+\sin(2t))^2$; $x(t) = (2+0{,}5\sin(2t))^{1/2}$; $x(t) = te^{-t}\cos(2\pi t+\pi/4)$;

$x(t) = 1.5 + 0.7t - 0.05t^2 - (e^{-3t}+t^3)^{1/3}$

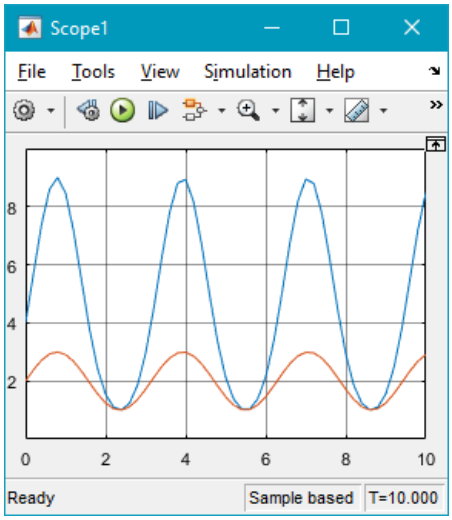
На рисунке 1.9 приведен один из вариантов построения сигнала: $x(t) = (2+sin(2t))^2$ с помощью функциональных блоков и результаты.



а



б



с

Рисунок 1.9 – Структурная схема модели сигнала $x(t) = (2+sin(2t))^2$ – *а* и графики двух сигналов на экране виртуального осциллографа: $2+sin(2t)$ –*б*, и $x(t) = 2+\sin(2t)$ – *с*

Задание 1.3. Моделирование интегрированного сигнала

В системах связи помехи и замирания, воздействующие на сигнал при его прохождении по каналу связи, имеют статистический характер и могут быть описаны при помощи различных законов распределений. В частности, замирания в канале связи при отсутствии прямой видимости между абонентом и базовой станцией имеют рэлеевский закон распределения; аддитивные помехи (шумы) часто описываются нормальным (гауссовским) законом распределения; временные интервалы между вызовами в сетях связи обычно имеют экспоненциальный закон распределения и.т.д.

Построить модель для имитации смеси сигналов:

- синусоидального сигнала и белого шума (рисунок 1.10);
- синусоидального сигнала и гауссова шума (с математическим ожиданием m=0,и дисперсией σ^2 =0.01);
- синусоидального сигнала и шума с равномерным распределением.

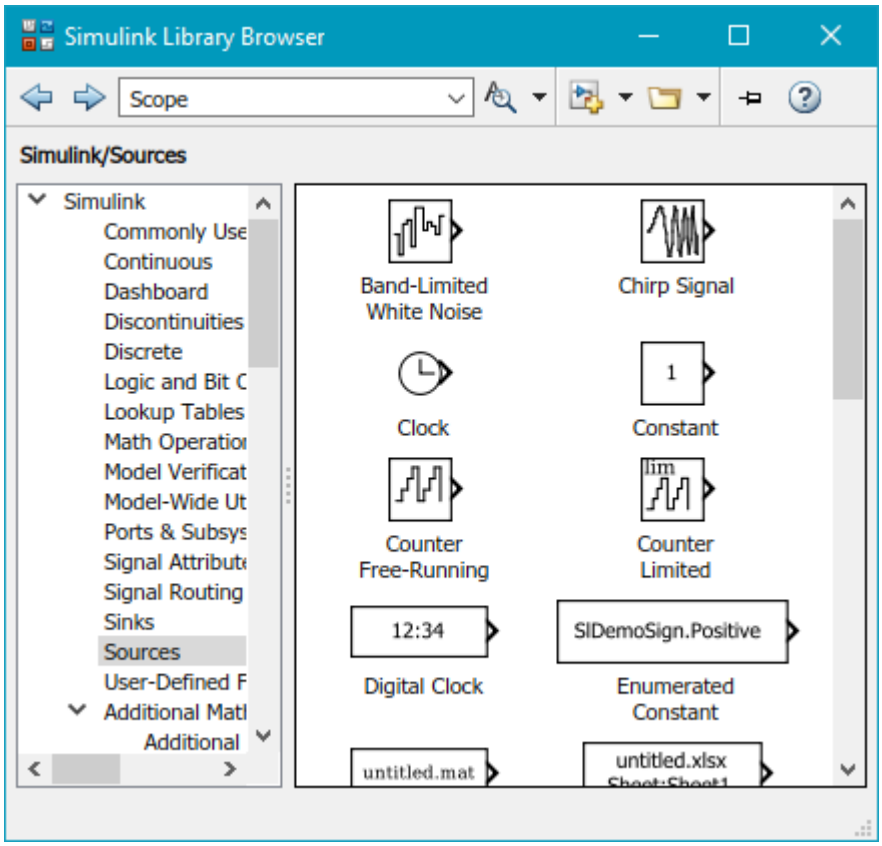


Рисунок 1.10 – Блоки библиотеки Sources

Пример построения модели для имитации смеси синусоидального сигнала белого шума и график полученного в результате имитации результирующего сигнала представлены на рисунках 1.11, 1.12.

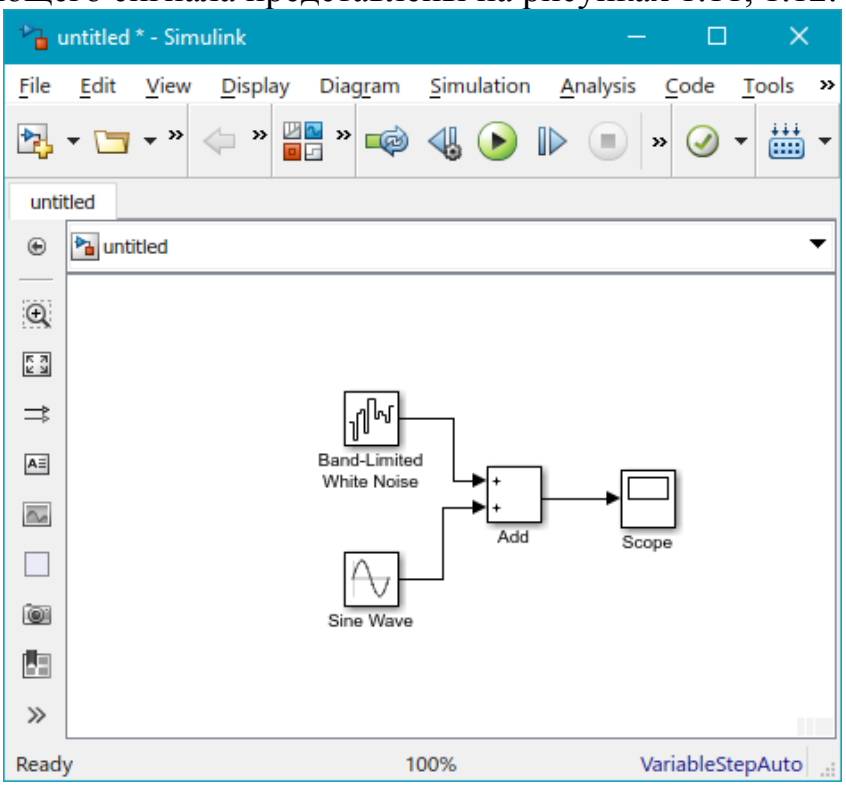


Рисунок 1.11– Модель имитации смеси сигналов синусоидального и белого шума

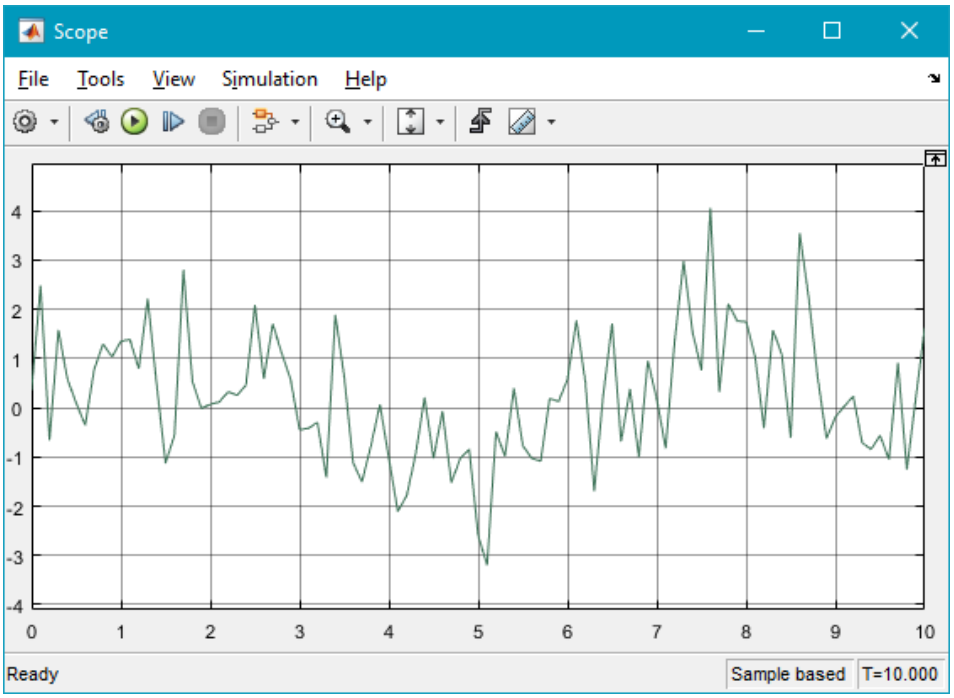


Рисунок 1.12 –Результат имитации смеси сигналов синусоидального и белого шума

Содержание отчета и его форма

Подготовьте отчет, в котором приведите технологию выполнения заданий.

Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) формулировку задания и технологию его выполнения;
4) ответы на контрольные вопросы;
5) приложение – файлы выполненных заданий.

Вопросы для защиты работы

1. Для чего используется система MATLAB?
2. Как осуществляется программирование в среде MATLAB?
3. Для чего используется пакет Simulink?
4. Как с помощью Simulink можно моделировать поведение сложных систем?
5. На какой технологии основана разработка моделей средствами SIMULINK (S-модели)?

ЛАБОРАТОРНАЯ РАБОТА 2
Моделирование объектов, описываемых системами уравнений

Цель и содержание: использование классических методов аналоговой вычислительной техники для моделирования объектов, описываемых системами алгебраических уравнений, приобретение навыков построения таких моделей в системе компьютерной математики MATLAB.

Организационная форма занятий: решение проблемных задач, разбор конкретных ситуаций

Вопросы для обсуждения на лабораторном занятии: построение моделей объектов, описываемых системами алгебраических уравнений.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Приближенное решение широкого круга вычислительных задач осуществляется с помощью методов решения систем линейных уравнений, если удается свести задачу к системе уравнений. Эти методы обычно делят на две большие группы. К первой группе относятся точные методы, которые позволяют для любых систем найти точные значения неизвестных после конечного числа точно выполняемых арифметических операций.

Ко второй группе относят приближенные методы, которые являются итерационными, так как решения в них получают в результате процесса приближений. Точные методы применяются для задач небольших размерностей (~10^2), а для задач большой размерности используют итерационные методы.

Особое место среди них занимают вероятностные методы, которые полезны лишь в случаях очень высокой размерности систем.

Для моделирования объектов или процессов, протекание которых можно описать с помощью системы линейных уравнений, применяют метод сущность которого заключается в замене системы линейных уравнений (2.1) системой дифференциальных уравнений (2.2) ей эквивалентной.

Пусть система линейных уравнений, описываемая исследуемый объект или процесс имеет вид:

$$\begin{aligned} a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n&=b_1\\ a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n&=b_2\\ &\ldots\\ a_{n1}x_1+a_{n2}x_2+\ldots+a_{nn}x_n&=b_n \end{aligned} \qquad (2.1)$$

или в матричном виде: $\boldsymbol{A}x=\boldsymbol{b}$, где $\boldsymbol{A}$–квадратная матрица размером $n \times n$, $\boldsymbol{b}$ и x – векторы размером n (n– размерность системы).

Заменим данную систему алгебраических уравнений эквивалентной системой дифференциальных уравнений:

$$\begin{aligned}
&\frac{dx_1}{dt}+a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n-b_1=0\\
&\frac{dx_2}{dt}+a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n-b_2=0\\
&\qquad\ldots\\
&\frac{dx_n}{dt}+a_{n1}x_1+a_{n2}x_2+\ldots+a_{nn}x_n-b_n=0
\end{aligned} \qquad (2.2)$$

Доказательством эквивалентности систем 2.1 и 2.2 является наличие затухающего решения системы дифференциальных уравнений 2.2, т.е. как только все производные станут равны $\left(\frac{dx_i}{dt}=0\right)$, решение системы уравнений 2.1 будет найдено.

Для обеспечения затухающего решения достаточным условием является положительное значение определенности матрицы коэффициентов системы уравнений. Это возможно, в частности, при условии, когда

$$a_{ij}\geq\sum_{j=1}^{n}a_{ij}, i\neq j. \qquad (2.3)$$

Рассмотрим пример решения системы линейных алгебраических уравнений:

$$\begin{cases}4x_1+2x_2=14\\2x_1+5x_2=-5\end{cases} \qquad (2.4)$$

с помощью инструментальных средств табличного процессора Excel; систем Mathcad и Matlab, пакета Simulink.

Рассмотрит технологию решения системы (2.4) с помощью выше указанных программных средств.

Решение системы линейных алгебраических уравнений в табличном процессоре Excel можно получить, используя матричную форму записи системы линейных уравнений в: ***Ax = b***, следовательно, вектор решения: x $=\boldsymbol{A^{-1}\cdot b}$. На рисунке 2.1. представлено решение системы уравнений 2.4 с помощью Excel. Алгоритм решения приведен работе №4

Решение систем линейных уравнений в MathCAD можно получить с помощью встроенной функции **lsolve (*A*,*b*)**, которая возвращает вектор ***x*** для системы линейных уравнений при условии, если задана матрица коэффициентов и вектор свободных членов. Также для решения системы линейных уравнений используются формулы Крамера и метод Гаусса.

На рисунке 2.2 представлено решение системы линейных уравнений методом обратной матрицы и с помощью функции **lsolve(*A*,*b*)**.

Для решения системы уравнений по формулам Крамера необходимо:

1. Переменной ORIGIN присвоить значение равное единице.
2. Ввести матрицу системы и столбец правых частей системы уравнений.
3. Вычислить определитель матрицы системы, учитывая, что система имеет единственное решение, если определитель отличен от нуля.

4. Вычислить определители новых матриц, полученных путем замены соответствующего столбца столбцом свободных членов.

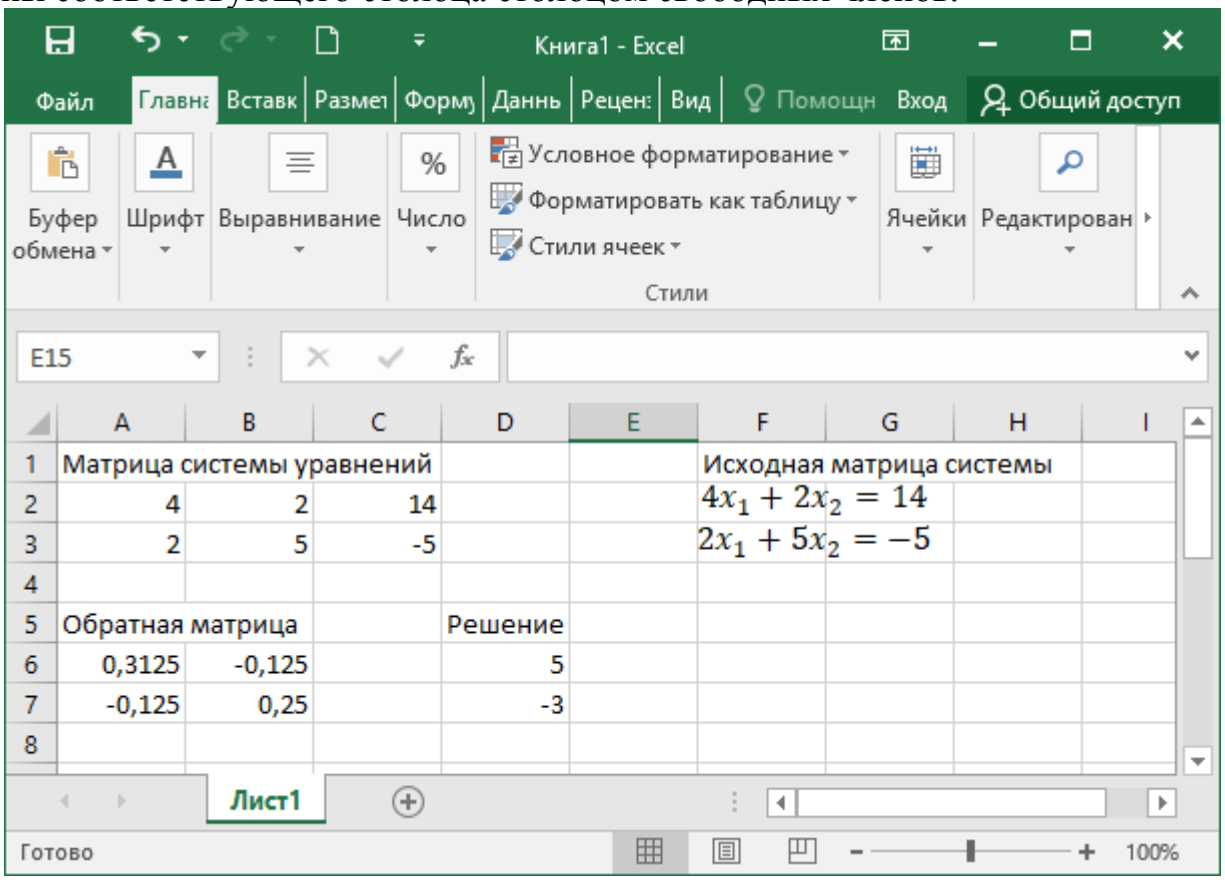


Рисунок 2.1 – Рабочий лист с решением системы линейных алгебраических уравнений (2.4)

$4 \cdot x_1 + 2 \cdot x_2 = 14$

$2 \cdot x_1 + 5 \cdot x_2 = -5$

Запись системы линейных уравнений в матричном виде

$A := \begin{pmatrix} 4 & 2 \\ 2 & 5 \end{pmatrix} \qquad b := \begin{pmatrix} 14 \\ -5 \end{pmatrix}$

Решение системы линейных уравнений

$x := A^{-1} \cdot b \qquad x = \begin{pmatrix} 5 \\ -3 \end{pmatrix}$

Решение системы линейных уравнений с помощью функции lsolve(A,b)

$x := \text{lsolve}(A, b) \qquad x = \begin{pmatrix} 5 \\ -3 \end{pmatrix}$

Рисунок 2.2 – Решение системы линейных уравнений 2.4 методом обратной матрицы и с помощью функции **lsolve (A,b)**

$ORING := 1$

$A := \begin{pmatrix} 4 & 2 \\ 2 & 5 \end{pmatrix} \qquad b := \begin{pmatrix} 14 \\ -5 \end{pmatrix} \qquad \Delta := |A| \qquad \Delta = 16$

$\Delta 1 := \begin{pmatrix} 14 & 2 \\ -5 & 5 \end{pmatrix} \qquad \Delta 2 := \begin{pmatrix} 4 & 14 \\ 2 & -5 \end{pmatrix}$

$|\Delta 1| = 80 \qquad |\Delta 2| = -48$

$x_1 := \frac{|\Delta 1|}{\Delta} \qquad x_2 := \frac{|\Delta 2|}{\Delta}$

$x_1 = 5 \qquad x_2 = -3$

Рисунок 2.3 – Рабочий лист Mathcad с решением системы уравнений 2.4 методом Крамера

5. Определить решение системы с помощью формул Крамера: $x_i = \frac{\Delta_i}{\Delta}$, где Δ – определитель матрицы системы, Δ_i – определитель матрицы *n*-го порядка, полученный из матрицы системы заменой *i*-го столбца столбцом правых частей системы уравнений.

Решение системы линейных уравнений с помощью формул Крамера представлено на рисунке 2.3.

Для решения системы уравнений методом Гаусса необходимо:

1. Переменной **ORIGIN** присвоить значение равное единице.
2. Ввести матрицу системы и столбец правых частей системы уравнений.
3. Сформировать расширенную матрицы системы с помощью функции **augment**(***A***,***b***).
4. Привести расширенную матрицу системы к ступенчатому виду с помощью функции **rref**(***Ar***).
5. Вывести решение системы, выделив последний столбец полученной матрицы для чего сформировать столбец решения системы с помощью функции **submatrix** (***Ag***, *ir*, *jr*, *ic* ,*jc*) – блок матрицы ***Ag***, состоящий из всех элементов, содержащихся в строках от *ir* до *jr* и столбцах от *ic* до *j*c.
6. Проверить правильность решения вычислив ***Ax– b***.

Выполните решение системы уравнений методом Гаусса самостоятельно.

На рисунке 2.4 представлен результат решения системы уравнений (2.4), в MATLAB.

Рисунок 2.4 – Решение системы уравнений (2.4) в MATLAB

Для решения данной системы в пакете Simulink необходимо перейти к эквивалентной системе дифференциальных уравнений в соответствии с системой (2.2):

$$\begin{cases} \frac{dx_1}{dt} = 14 - 4x_1 - 2x_2 \\ \frac{dx_2}{dt} = -5 - 2x_1 - 5x_2 \end{cases} \quad (2.5)$$

Структурная схема модели данной системы приведена на рисунке 2.5. Переходный процесс установления решения изображен на экране виртуального осциллографа (рисунок 2.6).

На рисунке 2.6 видно, что после $t = 2$ на выходах виртуальных интеграторов устанавливаются сигналы, соответствующие решению системы линейных алгебраических уравнений:

$$x_1 = 5,$$
$$x_2 = -3.$$

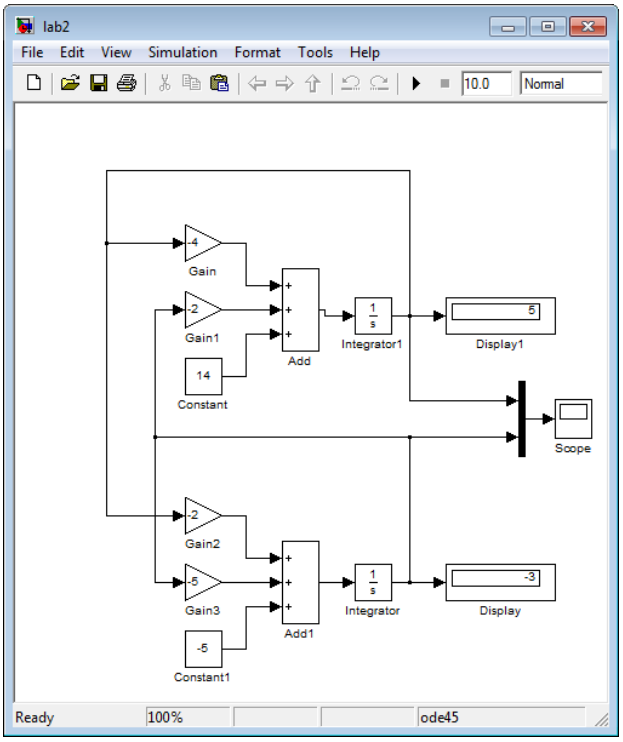


Рисунок 2.5 – Структурная схема модели системы дифференциальных уравнений 2. 6

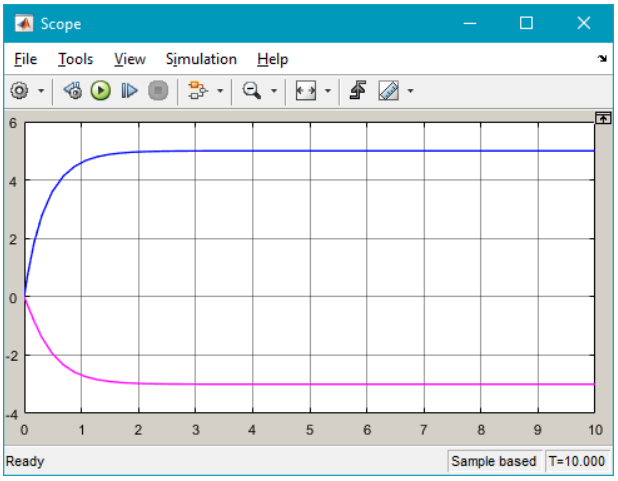


Рисунок 2.6 – Переходный процесс установления решения системы линейных алгебраических уравнений [2]

Аппаратура и материалы. Для выполнения лабораторной работы необходимо использовать следующее: аппаратное обеспечение: персональный компьютер и программное обеспечение: операционную систему Windows 10 и выше; Microsoft Office13 и выше, системы компьютерной математики Machcad и MATLAB R2017a и выше.

Указания по технике безопасности. Студенты должны следовать общепринятой технике безопасности для пользователей персональных компьютеров. Не следует самостоятельно производить ремонт технических средств, установку и удаление программного обеспечения. В случае обнаружения неисправностей необходимо сообщить об этом администратору компьютерного класса.

Методика и порядок выполнения работы

Выполните все примеры, приведенные в теоретической части и предложенные задания, предварительно ознакомившись с теоретической частью.

Задание 2.1. Построить модель решения задачи. Отряд туристов вышел в поход на нескольких байдарках ***L***, часть из которых ***A***-местные, а часть –***B***-местные. Сколько ***A***-местных и сколько ***B***-местных байдарок было в походе, если отряд состоит из ***M*** человек?

Переходный процесс установления решения выведите на экран виртуального осциллографа. Параметры сигналов приведены в таблице 2.1. Номер варианта соответствует номеру, под которым студент записан в списке группы.

Таблица 2.1– Параметры задачи

№ вар-та	A	B	L	M
1.	1	2	12	20
2.	1	3	7	17
3.	1	4	5	11
4.	1	5	8	16
5.	2	3	8	19
6.	2	4	10	37
7.	2	5	8	16
8.	3	4	8	26
9.	3	5	6	24
10.	4	5	7	33

Задание 2.2.Построить модель системы линейных алгебраических уравнений вида:

$$A x_1 + B x_2 + C x_3 = D$$
$$L x_1 + M x_2 + N x_3 = O$$
$$Q x_1 + R x_2 + S x_3 = T$$

Переходный процесс установления решения выведите на экран виртуального осциллографа. Параметры сигналов приведены в таблице 2.2. Номер варианта соответствует номеру, под которым студент записан в списке группы.

Таблица 2.2 – Параметры сигнала

№ вар-та	A	B	C	D	L	M	N	O	Q	R	S	T
1.	5	2	1	2.5	3	7	2	-1.5	4	0.5	5	11.5
2.	2	1.5	3.5	-16.25	2	3	4	-19.5	2	-0.5	5	-22.75
3.	0.5	-8	-9	-30.5	4	5	6	25.5	-1.5	2	3	8.5
4.	-1	2.5	9	-35.25	5	6	7	-22	1	2	-2.5	10
5.	3	-2	-7	-8.5	6	7	8	38	1	-3.5	6	32
6.	4.5	5.5	4	-3.25	7	8	9	-12	-4.5	5	1	38
7.	6.5	-5	-6	39	1	2	3	10.5	4	1	-5.5	7.25
8.	5	7.5	9.5	27.75	2	3	4	-4.5	1	-6.5	9	11.75
9.	-4	10	-3	-67.5	3	4	5	15.5	-7.5	8	1	-90.5
10.	8	1	6	64.5	4	5	6	22.5	7	1	-8.5	70

Содержание отчета и его форма

Подготовьте отчет, в котором приведите технологию выполнения заданий.

Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) формулировку задания и технологию его выполнения;
4) ответы на контрольные вопросы;
5) приложение – файлы выполненных заданий.

Вопросы для защиты работы

1. Как с помощью Simulink можно моделировать объекты, описываемые системами алгебраических уравнений?
2. Какие основные математические блоки используются для моделирования объектов, описываемых системами алгебраических уравнений?
3. Что является достаточным условием, обеспечивающим затухающее решение системы дифференциальных уравнений?

ЛАБОРАТОРНАЯ РАБОТА 3
Моделирование с помощью нейронных сетей. Основы байесовских методов классификации

Цель и содержание работы: познакомить студента с основами байесовских методов классификации.

Организационная форма занятий: решение проблемных задач, разбор конкретных ситуаций

Вопросы для обсуждения на лабораторном занятии: основные модели теории нейронных сетей, построение простейших моделей нейронных сетей в MATLAB.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Изучите теоретическое обоснование, используя информацию представленную ниже и соответствующую литературу.

В настоящее время искусственные нейронные сети применяются для решения многих не формализуемых или трудно формализуемых задач, таких как:

Задача обработки сигналов и изображений.

При обработке изображения (сигнала) часто необходимо отделить полезную (информативную) компоненту сигнала от шума. Такие задачи получили название задач фильтрации. Задачи подобного рода важны, в частности, в полиграфии. Например, нужно выделить наиболее существенную для пользователя информацию и убрать все остальное. Это сложная задача теории искусственного интеллекта.

Задача распознавания образов. Распознавание речи.

Пусть, например, на конвейере движется продукция. Необходимо создать систему технического зрения, позволяющую автоматически различить бракованную и нормальную продукцию. Другой пример: курсы валют, акций обычно сильно колеблются. Эти колебания связаны с тем, что на рынке имеется много агентов, действующих, как правило, несогласованно. Трейдер решает, в частности, такую задачу — выделить среди этих колебаний так называемый основной тренд, чтобы понять, стоит ли вкладывать свои деньги в акции. Основной тренд представляет собой некую усредненную кривую, где мелкие колебания (шум) должны быть устранены в результате фильтрации. Еще одна популярная ныне и крайне важная задача о выделении основного тренда — это задача анализа климатических данных с целью выяснения, есть глобальное потепление или нет.

Еще несколько важных задач: – распознавание спама в Интернете; автоматический перевод, распознавание почерка и так далее.

Нейронные сети можно использовать при следующих условиях:

1. Если задачу может решить человек.

2. Если при решении задачи можно выделить множество входных факторов (сигналов, признаков, данных и т.п.) и множество выходных факторов.

3. Если изменения входных факторов приводит к изменению выходных.

В то же время применение нейронных сетей при решении некоторых задач может оказаться эффективнее использования разума человека. Это объясняется тем, что человеческий разум ориентирован на решение задач в трехмерном пространстве. Многомерные задачи для него характеризуются значительно большей трудоемкостью.

Искусственным нейронным сетям не свойственно такое ограничение. Им все равно решать трехмерную или 10-мерную задачу.

Подобно биологической нейронной системе ИНС является вычислительной системой с огромным числом параллельно функционирующих простых процессоров с множеством связей.

Модели ИНС в некоторой степени воспроизводят "организационные" принципы, свойственные мозгу человека. Моделирование биологической нейронной системы с использованием ИНС может также способствовать лучшему пониманию биологических функций. Такие технологии производства, как VLSI (сверхвысокий уровень интеграции) и оптические аппаратные средства, делают возможным подобное моделирование. [Богатиков, В.Н. Построение систем управления на основе нейронных сетей: Учебно–методическое пособие / В.Н. Богатиков, Л.В. Дранишников, А.Е. Пророков. - Апатиты: Изд-во КФ ПетрГУ, 2011. – 41 с]

Общие принципы функционирования нейронных сетей [Вакуленко С.А., Жихарева А.А. Практический курс по нейронным сетям – СПб: Университет ИТМО, 2018. – 71 с.]

В основе построения нейронных моделей лежит использование аналогии с мозгом и реальными нейронами. Мозг состоит из нейронов, нейроны связаны друг с другом и обмениваются сигналами. Сигналы имеют булевскую природу, а система связанных нейронов – это стохастическая динамическая система.

Таким образом, нейрон – это пороговая система, которая получает входные сигналы от других нейронов, суммирует их и генерирует выходной сигнал, если эта сумма превышает некий порог.

Идеализированная модель такой пороговой системы может быть построена при помощи сигмоидальных функций, например, функции:

$$¿\frac{1}{1+e^{ax}}, \tag{3.1}$$

где x – амплитуда входного сигнала, который получает нейрон от других нейронов; σ – выходной сигнал нейрона, представленной на рисунке 3.1, или функции Хевисайда:

$$¿\begin{cases}1, x>0\\0, x\le 0\end{cases}, \qquad (3.2)$$

представленной на рисунке 3.2, или кусочно-линейная функция σ = max(*x*,0).

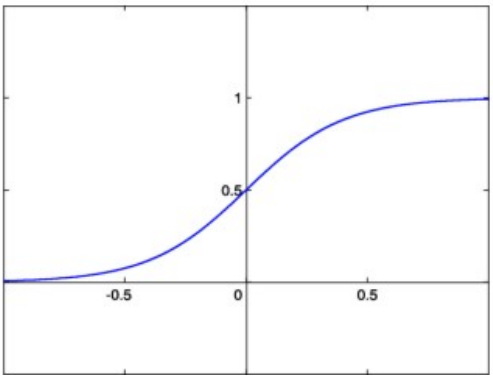


Рисунок 3.1 – График сигмоидальной функции ¿ $\frac{1}{1+e^{ax}}$

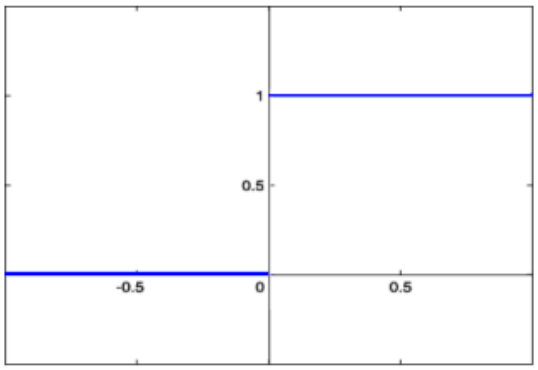


Рисунок 3.2 – График функции Хевисайда

Для решения задачи распознавания образов можно использовать рекуррентную нейросеть – многослойный персептрон. Силу связи между нейронами описывает матрица синаптических связей (*W*), в которой хранится вся долговременная фундаментальная информация. В процессе обучения мозга матрица синаптических связей медленно меняется со временем. Таким образом, можно разделить два динамических процесса, протекающих в мозге: быструю динамику нейронов, которая описывается динамической системой, и медленную, связанную с изменением силы связи между нейронами. Медленная динамика упрощенно описывается правилом Хебба (Дональд Хебб – канадский физиолог, экспериментально показавший, что если два нейрона одновременно активны, то сила связи между ними растет).

Рассмотрим общие принципы функционирования нейронных сетей на примере задачи классификации.

Предположим, мы хотим создать автоматическую систему, которая различает два типа объектов – *A* и *B*. Системы технического зрения позволяют записать данные об объектах (признаки объекта) в цифровом виде.

Предположим, что мы характеризуем объект с помощью набора признаков $(x_1, x_2, \ldots, x_n)$. Признаки могут быть выражены с помощью целых чисел или даже булевских переменных, то есть «есть признак» – «нет признака».

Совокупность признаков можно рассматривать как вектор с n компонентами, или точку *k*-мерного Эвклидова пространства $X=(x_1, x_2, \ldots, x_n)$. Тогда задача классификации сводится к следующей математической задаче: разделить два множества точек *A* и *B* *n*-мерного эвклидова пространства некоторой гиперповерхностью размерности *n*-1.

Сделаем некоторые важные комментарии. Выбор признаков для классификации является исключительно сложной задачей, которую мы

пока не рассматриваем. Ранее эта задача решалась вручную, в последнее время появились эффективные методы автоматического нахождения наиболее эффективных признаков (так называемый Deep Learning). Отметим, что выбор правильных признаков важен для успешности последующей обработки системы признаков методами, которые мы описываем ниже. Обучение нейронной сети в задачах классификации происходит на наборе обучающих примеров $X(1),X(2),\ldots,X(P)$, для которых принадлежность объекта к классу A или классу B известна. Чтобы математически формализовать этот факт, определим индикатор:

$$D X=\begin{cases}1, X A\\0, X B\end{cases}, \tag{3.3}$$

По накопленному в результате обучения «опыту» строим нейросеть (систему ИИ), которая проводит разделяющую поверхность.

Математически этот процесс может быть описан как поиск некоторой функции $y = F(X,W)$, где W – набор параметров нейронной сети (или другой системы ИИ). Для нейросетей эти параметры, в частности, задают силу связи между нейронами и подбираются так, чтобы ошибка обучения (*error training*) была бы минимальной (как можно ближе к нулю). В качестве ошибки обучения обычно рассматривают функцию

$$E_{train}(W)=\left|F X j W-D X j\right|, \tag{3.4}$$

где $X(j)$, $j\overline{1,P}$ берутся из обучающего множества.

Для проверки эффективности обучения нейронной сети берут тестовое множество объектов и вычисляют

$$E_{test}(W)=\left|F X jW-D X j\right|, \tag{3.5}$$

где $X(j)$ взяты из тестового множества.

После того, как система обучена (что иногда требует большого процессорного времени), она для любого поданного на вход системы объекта X автоматически решает, к какому классу он относится.

Рассмотрим основные модели теории нейронных сетей, выделив отдельно два разных класса сетей: ***Сети прямого распространения***: однослойный персептрон; многослойный персептрон; сети радиальных базисных функций (RBF);машины опорных векторов (SVM). ***Рекуррентные сети***: реккурентные сети; аттракторные нейронные сети и т.д.

Одна из первых моделей нейронной сети была предложена Ф. Розенблаттом в 50-е годы XX века и получила название однослойного персептрона. В этой модели входной сигнал проходил всего один слой нейронов и после этого формировался выходной сигнал при помощи сигмоидальной функции.

В дальнейшем выяснилось, что возможности таких простейших систем ограничены. Было предложено обобщение — модель многослойного персептрона, отличие которого от однослойного заключается в прохождении входного сигнала через несколько слоев.

Многослойные персептроны могут моделировать любую систему «вход-выход», то есть любой черный ящик. Однако они имеют ряд недостатков, в частности, выбор числа слоев, нейронов и связей между ними не так прост.

Такие системы, как сети радиальных базисных функций и машины опорных векторов, можно рассматривать как специальные модификации многослойных персептронов с целью преодоления недостатков последних.

Все перечисленные модели характеризуются наличием слоев, через которые проходит сигнал. В рекуррентных нейронных сетях невозможно выделить отдельные слои. Сигналы могут циркулировать по сети во всех направлениях, образуя сложную пространственно-временную структуру (*pattern*). Такие сети обладают колоссальными возможностями. Так, например, известно, что они могут моделировать любую динамическую систему или машину Тьюринга, то есть реализовать любой алгоритм.

Рассмотрим однослойный персептрон, который представляет собой простейшую модель нейронной сети. Однослойный персептрон получает на вход сигнал, заданный вектором X и на выходе выдает число $y = \sigma(S)$, где $Sw\sum_k w_k x_k h w_1 x_1 w_2 x_2 \ldots w_n x_n h$. Входной вектор X состоит из n компонент $X = (x_1, x_2, \ldots, x_n)$, каждая из которых представляет собой численную характеристику анализируемого нейронной сетью объекта. Через $w_i, i\overline{1,n}$ и h обозначены параметры персептрона — синаптические веса и порог (сдвиг) соответственно.

Синаптические веса указывают силу влияния конкретной характеристики на выходное значение. Значения этих параметров могут быть как положительными, так и отрицательными. Отметим, что переменные $x_i, i\overline{1,n}$ и могут быть булевскими, то есть принимать значения 0 или 1. Это означает, что объект обладает данным признаком, если $x_i = 1$, и не обладает, в случае, если $x_i = 0$. В этом случае функция $(S) = H S = \begin{cases} 1, S > 0 \\ 0, S \le 0 \end{cases}$ – функция Хевисайда или функция скачка. На рисунок 3.3 приведена модель однослойного персептрона, цифрами 1-4 помечена последовательность действий при работе алгоритма.

Преимущество персептрона заключается в его простоте. Укажем и его недостаток — персептрон классифицирует только линейно разделимые объекты (то есть только такие множества векторов $X(t)$, между которыми можно провести разделяющую эти множества гиперплоскость).

Геометрическая интерпретация Однослойный персептрон работает как классификатор объектов, которые математически могут быть заданы как элементы двух разных множеств в многомерных линейных пространствах. Чтобы понять, как это происходит, рассмотрим случай, когда сигмоидальная функция есть функция скачка $\sigma = H(S)$. Предположим, требуется классифицировать объекты в два непересекающихся класса объектов A и B.

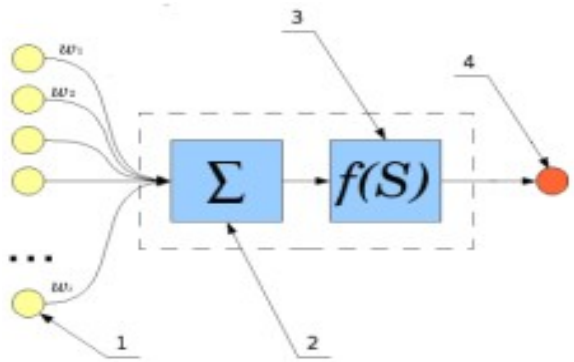


Рисунок 3.3 – Модель однослойного персептрона

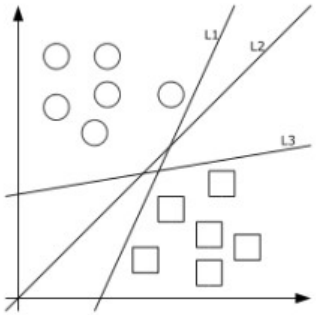


Рисунок 3.4. Геометрическая интерпретация однослойного персептрона

Для этого, определим выходные значения: положим $D = 1$, если объект принадлежит к классу A, $D = 0$, если объект принадлежит к классу B: $X = \begin{cases} 1, X A \\ 0, X B \end{cases}$. Тогда, из определения функции скачка $\sigma = H(S)$ следует, что объект лежит в классе A при условии $S > 0$ и объект лежит в классе B, если $S < 0$. Следовательно, уравнение $S = 0$ может быть рассмотрено как некий разделитель множеств объектов разных классов. С другой стороны, уравнение $S = 0$ равносильно уравнению $w_1 x_1 w_2 x_2 \ldots w_n x_n h = 0$, которое определяет гиперплоскость в n-мерном линейном пространстве, состоящем из векторов (строк) $X = (x_1, x_2, \ldots, x_n)$

Напомним, что в двумерном пространстве гиперплоскость — это прямая, а в трехмерном — обычная плоскость. Иллюстрация на рисунке 3.4 демонстрирует вышесказанное на примере игры в квадратики (класс A) и шарики (класс B) в случае n=2, то есть на плоскости. Прямые L_1, L_2, L_3 – примеры «разделителей» этих классов. Таким образом, однослойный персептрон есть линейный разделитель.

Моделирование логических функций однослойным персептроном Однослойный персептрон позволяет несложным образом реализовать ряд логических функций, таких как конъюнкция, дизъюнкция, отрицание, демократическое голосование.

Результат каждой из этих функций определяется значением двух булевских переменных x_1 и x_2.

Представим значения этих переменных как вектор входных значений

$X = (x_1, x_2)$, а результат как значение функции скачка, то есть y= $H(S) = H(w\ x\ w\ x - h) = = 1\ 1 + 2\ 2 -$. Поясним сказанное на примере функции конъюнкции — булевской функции, определенной следующей таблицей:

X_1	0	1	0	1
X_2	0	0	1	1
$X_1 \wedge X_2)$	0	0	0	1

Наша задача — представить эту логическую функцию с помощью однослойного персептрона. Рассмотрим вектор входных значений $X = (x_1, x_2)$ как точку плоскости, а результат логической операции – как форму

точки: квадрат, в случае, когда результат операции истинен ($y = 1$), и круг, если результат операции ложен ($y = 0$). Тогда наша задача становится эквивалентной такой задаче классификации: найти прямую, разделяющую два класса объектов — квадраты и круги.

Прямой, разделяющей эти множества объектов, является, например, прямая $x_1 + x_2 - 1.5 = 0$. Это означает, что синаптические веса $w_1 = 1$, $w_2 = 1$, а порог $h = 1.5$. Иллюстрация вышесказанного приведена на рисунке 3.5.

Однако оказывается, что иногда квадраты и круги неразделимы. Марвин Минский показал, что возможности персептрона ограничены, потому что он разделяет множества с помощью гиперплоскости (пример Марвин). В самом деле, есть такие явно различимые классы объектов, для которых прямая как их разделитель не подходит. Например, рассмотрим логическую функцию «исключающее ИЛИ» (*XOR*), определяемую формулой $XOR(x, y) = x + y \pmod 2$. Для лучшего понимания приведем таблицу значений функции *XOR* для различных значений переменных:

X_1	0	1	0	1
X_2	0	0	1	1
$XOR\ (X_1, X_2)$	0	1	1	0

Эта функция не реализуема однослойным персептроном, то есть не существует такой прямой, которая бы разделила эти два класса объектов. На рисунке 3.6 проиллюстрирован результат этой логической операции в виде квадратов, в случае, когда результат операции истинен ($y = 1$), и круга, если результат операции ложен ($y = 0$). Очевидно, что эти два множества (квадраты и круги) не разделимы никакой прямой.

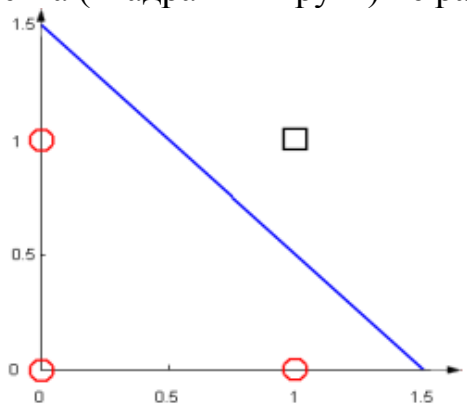


Рисунок 3.5 – Моделирование конъюнкции однослойным персептроном

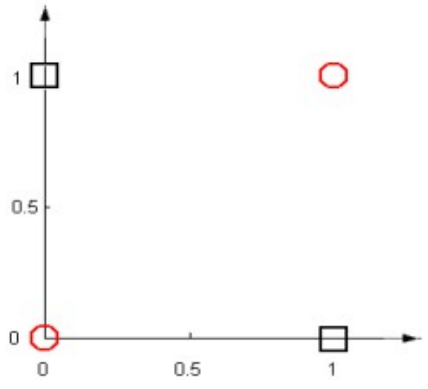


Рисунок 3.6 – Пример Минского

Алгоритм обучения однослойного персептрона

Имеются итерационные алгоритмы, которые находят параметры, если исходные объекты линейно разделимы.

Обучение нейронной сети в задачах классификации происходит на наборе обучающих примеров $X(1), X(2), \ldots, X(P)$, , в которых ответ – принадлежность к классу A или B, – известен. Определим индикатор D

следующим образом: положим $D(X) = 1$, если X из класса A, и положим $D(X) = 0$, если X из класса B, то есть

$$D(X)\begin{cases}1, X A,\\ 0, X B\end{cases} \quad (3.6)$$

где всякий вектор X состоит из n компонент: $X=(x_1, x_2, \ldots, x_n)$. Задача обучения персептрона состоит в нахождении таких параметров $w_1, w_2, \ldots, w_n$ и h , что на каждом обучающем примере персептрон выдавал бы правильный ответ, то есть

$$y X i D X i, i\overline{1, P} \ (2) \quad (3.7)$$

Интуитивно кажется очевидным, что если персептрон обучен на большом числе корректно подобранных примеров, и равенство (3.7) выполнено для почти всех $X i, i\overline{1, P}$, то в дальнейшем персептрон будет с близкой к единице вероятностью проводить правильную классификацию для остальных примеров. Этот интуитивно очевидный факт был впервые математически доказан (при некоторых предположениях) в основополагающей работе наших соотечественников В. Вапника и А. Червоненскиса еще в 1960-х годах. На практике, однако, оценки по теории Вапника–Червоненскиса иногда не очень удобны, особенно для сложных моделей нейронных сетей. Поэтому практически, чтобы оценить ошибку классификации, часто поступают следующим образом: множество обучающих примеров разбивают на два случайно выбранных подмножества, при этом обучение идет на одном множестве, а проверка обученного персептрона — на другом. Рассмотрим подробнее алгоритм обучения персептрона.

Шаг 1. Инициализация синаптических весов и смещения.

Значения всех синаптических весов модели полагают равными нулю: $w_i = 0, i\overline{1, n}$; смещение нейрона h устанавливают равным некоторому малому случайному числу. Ниже, из соображений удобства изложения и проведения операций будем пользоваться обозначением $w_0 = -h$.

Обозначим через $w_i(t), i=\overline{1, n}$вес связи от i-го элемента входного сигнала к нейрону в момент времени t.

Шаг 2. Предъявление сети нового входного и желаемого выходного сигналов.

Входной сигнал $X=(x_1, x_2, \ldots, x_n)$предъявляется нейрону вместе с желаемым выходным сигналом D.

Шаг 3. Адаптация (настройка) значений синаптических весов. Вычисление выходного сигнала нейрона.

Перенастройка (адаптация) синаптических весов проводится по следующей формуле: $w_i(t + 1) = w_i(t) + r \cdot (D(t) - y(t)) \cdot x_i(t)$, $i\overline{0, n1}$ где под $D(t)$ подразумевается индикатор, определенный равенством (3.6), а r – параметр обучения, принимающий значения меньшие 1.

Описанный выше алгоритм — это алгоритм градиентного спуска, который ищет параметры, чтобы минимизировать ошибку. Алгоритм

итеративный. Формула итераций выводится следующим образом. Введем риск

$$R(t)=\sum_{e=1}^{T}\left(D(t)-F(X(t),W(t))\right)^2, \qquad (3.8)$$

где суммирование идет по числу опытов (t – номер опыта), при этом задано максимальное число опытов – T. Подставим вместо F формулу для персептрона, вычислим градиент по w. В результате мы получим указанную выше формулу перенастройки весов.

Иллюстрация работы алгоритма

В процессе обучения вычисляется ошибка $(t)D(t)y(t)$. Построим график, показывающий, как меняется эта ошибка (рисунок 3.7) в ходе обучения сети и адаптации весов. На нем хорошо видно, что, начиная с некоторого шага, величина $\delta(t)$ равна нулю. Это означает, что персептрон обучен.

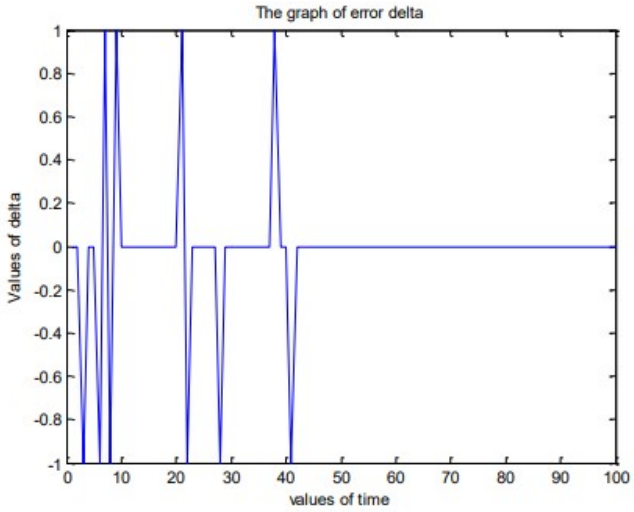


Рисунок 3.7 – График изменения ошибки в процессе обучения

Задача о моделировании черного ящика в общей постановке

Пусть известен реальный выход устройства $D(t)$ при входе $X(t)$, где $X(t)$ — вектор с компонентами $(x_1,x_2,\ldots,x_n)$, t – номер проводимого опыта $,t\overline{1,T}$ (максимальное число опытов T заранее определено). Необходимо найти параметры модели $w(w_0,w_1,\ldots,w_n)$, такие, что выход модели $F(X,w)$ и реальный выход устройства $D(t)$ были бы как можно ближе в среднеквадратичном смысле, то есть

$$R(t)=\sum_{e=1}^{T}\left(D(t)-F(X(t),W(t))\right)^2 \to min \qquad (3.8)$$

Функцию R(t) называют эмпирическим риском.

Аппаратура и материалы. Для выполнения лабораторной работы необходимо использовать следующее: аппаратное обеспечение: персональный компьютер и программное обеспечение: операционную систему Windows 10 и выше; Microsoft Office13 и выше, системы компьютерной математики Machcad и MATLAB R2017a и выше.

Указания по технике безопасности. Студенты должны следовать общепринятой технике безопасности для пользователей персональных компьютеров. Не следует самостоятельно производить ремонт технических средств, установку и удаление программного обеспечения. В случае обнаружения неисправностей необходимо сообщить об этом администратору компьютерного класса (обслуживающему персоналу лаборатории).

Методика и порядок выполнения работы

Выполните предложенные задания, предварительно рассмотрев и выполнив приведенные примеры.

Пример 1. Построение нейронной сети для операции конъюнкции в пакете Matlab

Построим нейронную сеть для логической операции конъюнкции с помощью пакета Matlab.

Таблица данной логической операции приведена выше. В данной реализации массив входов P задает значения X_1, X_2 для всех обучающих примеров и представляет собой первые две строки этой таблицы.

Целевой вектор T содержит значения индикаторной функции $D(X)$ для всех примеров X (в данном случае 4 примера) и представляет собой последнюю строку этой таблицы. На рисунке 3.8 представлен программный код и результат его выполнения в командном окне Matlab.

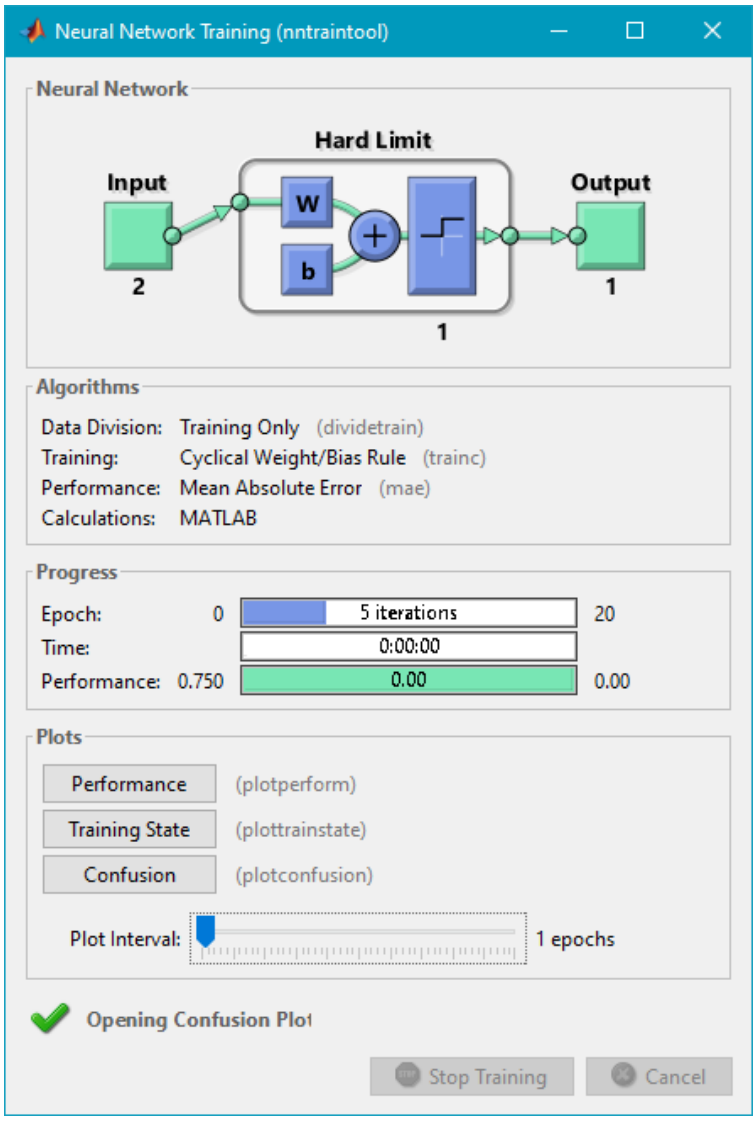
```
% аппроксимация конъюнкции однослойным персептроном
% построение функции y = H(w_1*X_1 + w_2*X_2 + w_0)
% (P,T) задают таблицу истинности
% зададим вектор входов
P = [ 0 1 0 1; 0 0 1 1 ];
% зададим вектор выходов
T = [ 0 0 0 1 ];
% встроенная функция построения сети с помощью персептрона,
% по умолчанию использована модель 'hardlim'
net = perceptron;
% определим максимальное число итераций для обучения сети
net.trainParam.epochs = 20;
% обучение сети
% в процессе обучения происходит коррекция весов w
net = train(net,P,T);
% симуляция
% вычисление значений аппроксимирующей функции в точках P
y = sim(net, P)
% синаптические веса после обучения сети
A = net.IW; celldisp(A)
% коэффициент сдвига b после обучения сети
w_0= net.b
```

```
Command Window
>> % аппроксимация конъюнкции однослойным персептроном
% построение функции y = H(w_1*X_1 + w_2*X_2 + w_0)
% (P,T) задают таблицу истинности
% зададим вектор входов
P = [ 0 1 0 1; 0 0 1 1 ];
% зададим вектор выходов
T = [ 0 0 0 1 ];
% встроенная функция построения сети с помощью персептрона,
% по умолчанию использована модель 'hardlim'
net = perceptron;
% определим максимальное число итераций для обучения сети
net.trainParam.epochs = 20;
% обучение сети
% в процессе обучения происходит коррекция весов w
net = train(net,P,T);
% симуляция
% вычисление значений аппроксимирующей функции в точках P
y = sim(net, P)
% синаптические веса после обучения сети
A = net.IW; celldisp(A)
% коэффициент сдвига b после обучения сети
w_0= net.b

y =

     0     0     0     1


A{1} =

     1     2



w_0 =

  cell

    [-3]
```

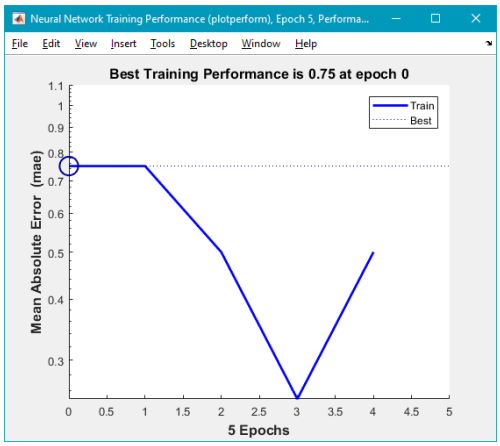
Рисунок 3.8 – Программный код и результат его работы в командном окне Matlab

В процессе работы программы Matlab отображает текущую информацию о модели и параметрах обучения в специальном окне Neural Network Training (рисунок 3.10). В частности, в этом окне показана структура нейронной сети — число входных нейронов (2), число уровней персептрона (1), число выходных нейронов (1), число итераций при обучении сети (в нашем случае — 5), время обучения сети.

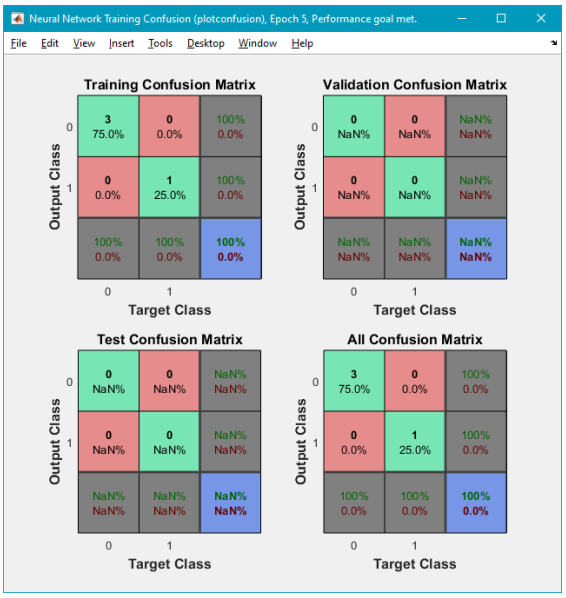
В то же время согласно инструкциям скрипта в окне команд будет выведена информация по результатам симуляции (вектор y) и значения весовых коэффициентов ($w_1 = 1$, $w_2 = 2$) и коэффициента сдвига ($w_0 = -3$):



а



б



с

Рисунок 3.10 – Нейронная сеть, реализующая конъюнкцию – а; настройка весов обучения – *б*; результат работы сети – с

Таким образом, получена модель персептрона, разделяющая классы объектов и определяемая уравнением $x_1 + 2x_2 - 3 = 0$, что геометрически будет означать разбиение двух множеств заданной прямой (рисунок 3.11).

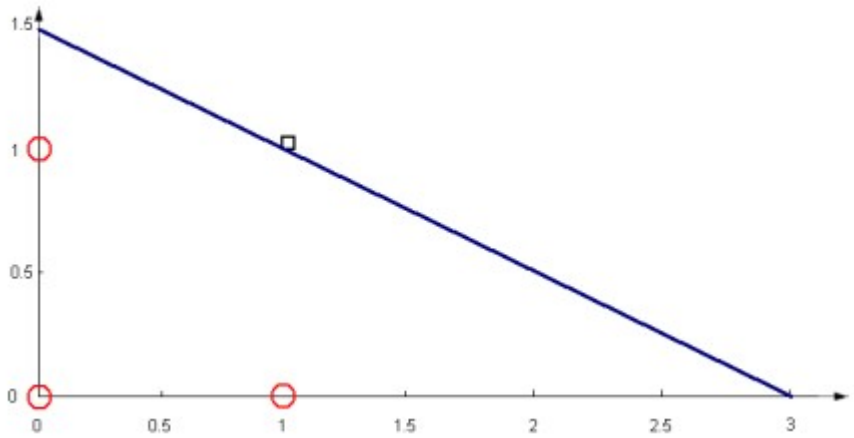


Рисунок 3.11– Моделирование конъюнкции однослойным персептроном в Matlab

Построение нейронной сети для классификации множеств точек плоскости в пакете Matlab

Пример 2. В качестве другого примера применения однослойного персептрона приведем пример классификации точек плоскости на два класса *A* и *B* – точки, находящиеся внутри эллипса, лежат в классе *A* и точки, находящихся вне эллипса, лежат в классе *B*. Эллипс определен общим уравнением кривой второго порядка вида:

$$a \cdot x^2 + b \cdot x \cdot y + c \cdot y^2 + d \cdot x + e \cdot y = 1 \qquad (3.9)$$

Ниже приведен программный код для эллипса, задаваемого уравнением:

$$1{,}5 \cdot x^2 + 2 \cdot x \cdot y + 1{,}1 \cdot y^2 + 2 \cdot x + 3 \cdot y = 1 \qquad (3.10)$$

Для наглядности приведены соответствующие построения на плоскости: исходная кривая, точки двух множеств, окрашенные соответственно в разные цвета.

Замечание. Этот пример показывает, что в исходном пространстве двух переменных однослойный персептрон не может разделить внешнюю и внутреннюю части эллипса. Однако можно сделать нелинейное преобразование данных, отобразив их в пространство высшей размерности. Здесь применяется так называемая теорема Ковера, которая утверждает, что после такого отображения данные могут стать линейно разделимыми.

Применим в нашем случае нелинейное отображение точки плоскости x, y в пространство размерности 5 следующим образом $: x, y = (a \cdot x^2 + b \cdot x \cdot y + c \cdot y^2 + d \cdot x + e \cdot y = 1)$

```
% построение однослойного персептрона для классификации
% точек плоскости в соответствии с заданным эллипсом
% (внутри и вне эллипса (ax^2 + bxy + cy^2 + dx + ey = 1))
% Результат работы нейронной сети:
% вектор значений t:
% 0 - точка вне эллипса, 1 - точка внутри эллипса
N = 300; % число точек для обучения
% коэффициенты заданного эллипса
a = 1.5; b = 2; c = 1.1; d = -2; e = -3;
```

```
% построим график исходного эллипса
syms X Y;
f = a*X.^2 + b*X.*Y + c*Y.^2 + d*X + e*Y-1;
ellips = ezplot(f); set (ellips,'Color','b'); hold on
% сгенерируем исходные данные для обучения:
% точки плоскости (x,y), чьи координаты имеют
% нормальное распределение на интервалах [-2a;2a] и [-2c;2c]
x = randn(1,N)*(2*a) - d/(2*a); y = randn(1,N)*(2*c) - e/(2*c);
% определим отображение
% (x1,x2)->(a*x1^2,b*x2^2,c*x1*x2,d*x1,e*x2)
% матрица входов
P = [a*x.^2; b*x.*y; c*y.^2; d*x; e*y];
% сформируем вектор выходов T, приписывая значения 0 и 1
% для точек, лежащих вне и внутри эллипса соответственно
T = ones(1,N);
k = find( sum(P) >= 1);
T(k) = 0;
% функция построения сети с помощью персептрона,
% по умолчанию использована модель 'hardlim'
net = newp(P,T);
% определим максимальное число итераций для обучения сети
net.trainParam.epochs = 50;
% функция обучения сети; при этом происходит коррекция весов W
net = train(net,P,T);
% симуляция
% вычисление значений аппроксимирующей функции в точках P
t = sim(net, P)
% синаптические веса после обучения сети
A = net.IW;
celldisp(A)
% коэффициент сдвига b после обучения сети
b = net.b
% геометрические построения
% распознавание точек плоскости после обучения сети
k = find( t == 0); % индексы точек вне эллипса
x_out = x(k); y_out = y(k);
% построение точек вне эллипса
plot(x_out, y_out, '*r');
clear k;
k = find( t == 1); % индексы точек внутри эллипса
x_out = x(k); y_out = y(k);
% построение точек внутри эллипса
plot (x_out, y_out, '*g')
```

В процессе работы программа отображает текущую информацию о модели и параметрах обучения в специальном окне Neural Network Training (рисунок 3.12). В частности, в этом окне показана структура нейронной сети – число входных нейронов (5), число слоев персептрона (1) и используемые модели слоев, число выходных нейронов (1), число итераций при обучении сети (в нашем случае — 50), время обучения сети

(в нашем случае – 4 с). В окне команд будут выведены значения весовых коэффициентов
(w_1= –122.6519, w_2= –132.2919, w_3= –137.2794, w_4= – 121.0269, w_5= – 127.4035) и коэффициента сдвига (w_0=151): A{1} –122.6519 –132.2919 –137.2794 –121.0269 –127.4035, b = [108], см. рисунок:

```
A{1} =

  -122.6519 -132.2919 -137.2794 -121.0269 -127.4035


b =

  cell

    [108]
```

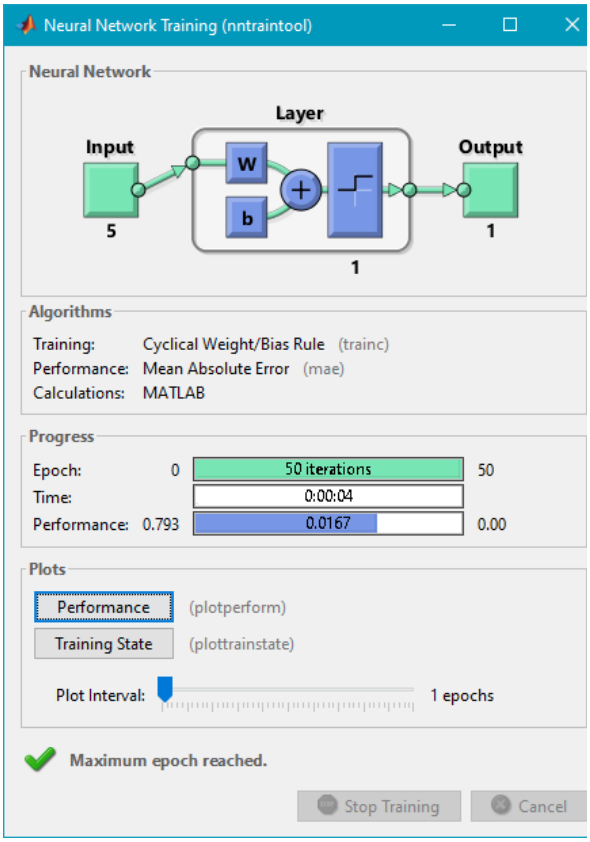
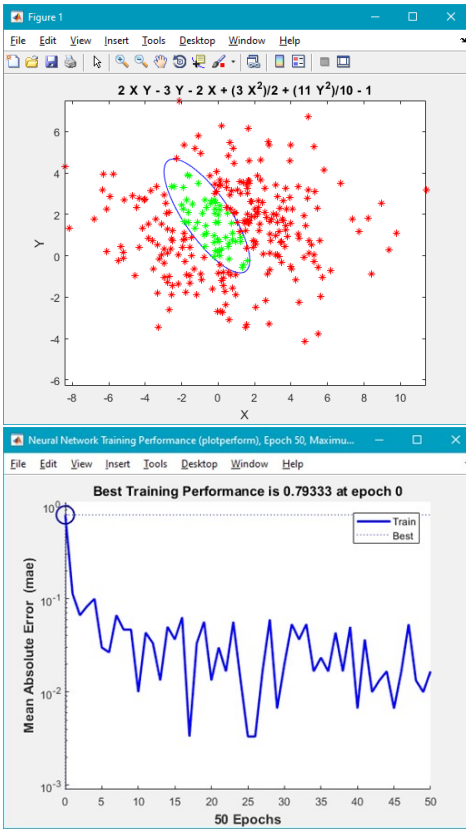


Рисунок 3.12. Однослойный персептрон



а

б

Рисунок 3.13. Классификация точек однослойным персептроном (*а*), график настройки весовых коэффициентов персептрона(*б*)

Таким образом, построенная модель имеет следующий вид (у всех весовых коэффициентов указано только два десятичных знака):

$$y = H(S) = H(w_1 x_1 + w_2 x_2 + \ldots + w_5 x_5 + w_0) = H - 122.65\,x_1 - 132.29\,x_2 - 137.28\,x_3 - 121.03\,x_4 - 127.40\,x_5 + 10$$

Результат классификации множества точек после обучения персептрона представлен графически на рисунке 3.13 б; точки «внутри» и «вне» эллипса изображены зеленым и красным цветом соответственно.

Рассмотрим байесовские методы классификации. Байесовский подход является классическим в теории распознавания образов и лежит в основе многих методов. Он опирается на теорему о том, что если плотности распределения классов известны, то алгоритм классификации, имеющий минимальную вероятность ошибок, можно выписать в явном виде. Для оценивания плотностей классов по выборке применяются различные подходы.

Рассмотрим пример параметрического подхода, но сначала определим вероятностную постановку задачи классификации.

Пусть X – множество объектов, Y – конечное множество имён классов, множестве X×Y является вероятностным пространством с плотностью распределения p(x, y) = P(y)p(x|y). Вероятности появления объектов каждого из классов P_y = P(y) называются априорными вероятностями классов. Плотности распределения $p_y(x)$ = p(x|y) называются функциями правдоподобия классов. Вероятностная постановка задачи классификации разделяется на две независимые подзадачи.

1. Имеется простая выборка $X^l = (x_i, y_i)_{i=1}^l$ (l – размер выборки) из неизвестного распределения p(x, y) = $P_y p_y(x)$. Требуется построить эмпирические оценки априорных вероятностей $\hat{P}_y$ и функций правдоподобия $\hat{p}_y(x)$ для каждого из классов $y \in Y$.

2. По известным плотностям распределения $p_y(x)$ и априорным вероятностям P_y всех классов $y \in Y$ построить алгоритм a(x), минимизирующий вероятность ошибочной классификации.

Первая задача имеет множество решений, поскольку многие распределения p(x, y) могли бы породить одну и ту же выборку X^l. Приходится привлекать различные предположения о плотностях, что и приводит к большому разнообразию байесовских методов.

В параметрическом подходе предполагается, что плотность распределения выборки известна.

Нормальный дискриминантный анализ – это специальный случай байесовской классификации, когда предполагается, что плотности всех классов $p_y(x)$, $y \in Y$ являются многомерными нормальными.

Приведём формулу *n*-мерного (гауссова) распределения:

$$p(x) = \frac{1}{(2\pi)^{n/2}|S|^{1/2}} \exp\left(-\frac{1}{2}(x-m)^T S^{-1}(x-m)\right), \quad (3.11)$$

где *n* – количество числовых признаков, $m = E[x]$ – математическое ожидание (центр), *S* – ковариационная матрица ($S = E[(x-m)(x-m)^T]$), |*S*| – детерминант *S*.

Пример 3. Вычислим, используя Matlab, значение гауссова распределения для $x_1 = [0.2,\ 1.3]^T$ и $x_2 = [2.2,\ -1.3]^T$, где $m = [0,\ 1]^T$, а $S = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Функцию нужно создать в отдельном файле Matlab (New → Function). Таким образом будет два файла Matlab, которые могут располагаться в одной папке. Чтобы Matlab понимал откуда ему загружать необходимую функцию, необходимо в окне Current Folder нажать правой кнопкой мыши на нужную папку, чтобы вызвать контекстное меню) и выбрать (Add to Path → Selected Folders and Subfolders), рисунок 3.14.

```
% Готовим Matlab к работе
close('all');
clear;
m=[0 1]'; S=eye(2);
x1=[0.2 1.3]'; x2=[2.2 -1.3]';
pg1=comp_gauss_dens_val(m,S,x1
)
pg2=comp_gauss_dens_val(m,S,x2
)
```

Где comp_gauss_dens_val (·) определим как

```
function
[z]=comp_gauss_dens_val(m,S,x)
[l,c]=size(m);
z=(1/(
(2*pi)^(l/2)*det(S)^0.5)      )*exp(-
0.5*(x-m)'*inv(S)*(x-m));
```

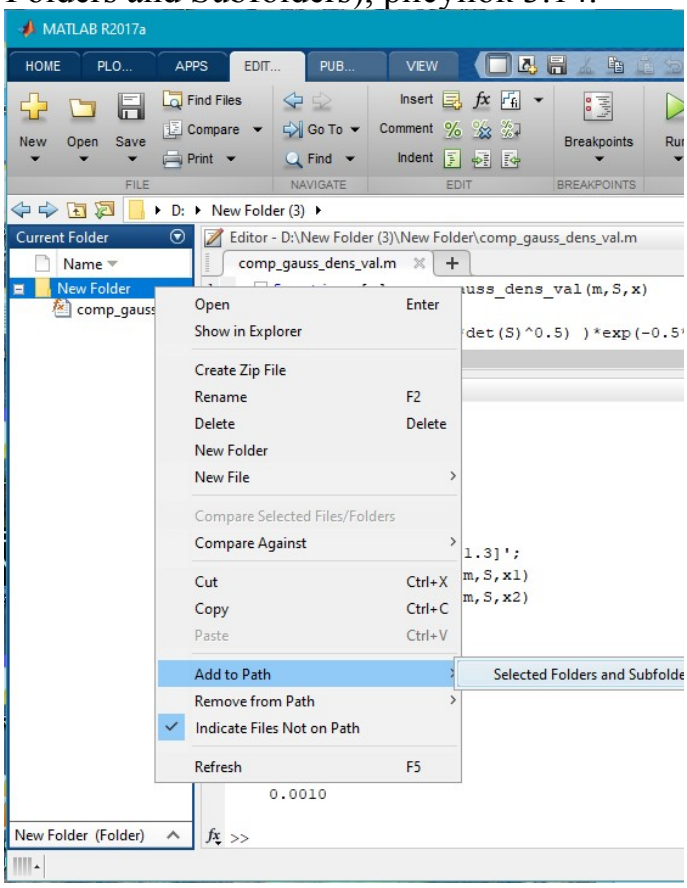


Рисунок 3.14 – Установка папки, откуда Matlab будет брать необходимые ему файлы

Функция **eye()** – создаёт единичную матрицу, запятая после квадратных скобок означает транспонирование вектора (**[]'**). Для получения справки по неизвестному объекту языка matlab, достаточно установить на него курсор и нажать F1 или написать команду help с именем объекта, допустим, **help eye**.

В результате выполнения кода получится, что pg1 = 0.1491, pg2 = 0.001.

На основании этого кода можно написать код байесовского классификатора, который определит к какому из двух классов относится входной вектор.

Например, пусть есть два класса w_1 и w_2, имеющих гауссовы

распределения $N(m_1, S_1)$, $N(m_2, S_2)$. $m_1 = [1, 1]^T$, $m_2 = [3, 3]^T$, $S_1 = S_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Примем во внимание, что вероятность появления первого и второго классов подсчитаны заранее и равны $P(w_1) = P(w_2) = ½$. Требуется узнать к какому классу принадлежит вход $x = [1.8, 1.8]^T$.

Пример 4. Напишем код байесовского классификатора, который решит эту задачу:

```
close('all');
clear;
 P1=0.5;
P2=0.5;
m1=[1 1]';
m2=[3 3]';
S=eye(2);
x=[1.8 1.8]';
p1=P1*comp_gauss_dens_val(m1,S,x)
p2=P2*comp_gauss_dens_val(m2,S,x)
```

```
>> close('all');
clear;
 P1=0.5;
P2=0.5;
m1=[1 1]';
m2=[3 3]';
S=eye(2);
x=[1.8 1.8]';
p1=P1*comp_gauss_dens_val(m1,S,x)
p2=P2*comp_gauss_dens_val(m2,S,x)

p1 =

    0.0420


p2 =

    0.0189
```

Получается, что $p_1 = 0.042$, $p_2 = 0.0189$, т.е. $p_1 > p_2$, соответственно двумерный вектор принадлежит первому классу.

Далее, проведём геометрический анализ нормальной плотности, её зависимость от матрицы ковариации.

Пример 5. Рассмотрим код, который генерирует 500 двумерных точек, распределенных в соответствие с двумерным нормальным распределением с центром $m = [0, 0]^T$ и матрицей ковариации $S = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$, где существует 8 типов этой матрицы:

1. $\sigma_1^2 = \sigma_2^2 = 1,\ \sigma_{12} = 0$;
2. $\sigma_1^2 = \sigma_2^2 = 0.2,\ \sigma_{12} = 0$
3. $\sigma_1^2 = \sigma_2^2 = 2,\ \sigma_{12} = 0$
4. $\sigma_1^2 = 0.2,\ \sigma_2^2 = 2,\ \sigma_{12} = 0$
5. $\sigma_1^2 = 2,\ \sigma_2^2 = 0.2,\ \sigma_{12} = 0$.
6. $\sigma_1^2 = \sigma_2^2 = 1,\ \sigma_{12} = 0.5$;
7. $\sigma_1^2 = 0.3,\ \sigma_2^2 = 2,\ \sigma_{12} = 0.5$.
8. $\sigma_1^2 = 0.3,\ \sigma_2^2 = 2,\ \sigma_{12} = -0.5$.

```
close('all');
clear;
% Генерируем точки для первого случая
randn('seed',0); % используем старый вариант вызова генератора
% случайных чисел, слово 'seed' заменяет ряд параметров для такого генератора
m=[0 0]'; % центр
S=[1 0;0 1]; % матрица ковариации
N=500; % количество точек
```

```
X = mvnrnd(m,S,N)'; % стандартная функция по генерации многомерного
% нормального случайного распределения
% Рисуем точки для первого варианта
figure(1), plot(X(1,:),X(2,:),'.');
figure(1), axis equal % устанавливаем тип стиля для осей
figure(1), axis([-7 7 -7 7])
% Рисуем точки для второго варианта
m=[0 0]';
S=[0.2 0;0 0.2];
N=500;
X = mvnrnd(m,S,N)';
figure(2), plot(X(1,:),X(2,:),'.');
figure(2), axis equal
figure(2), axis([-7 7 -7 7])
% Рисуем точки для третьего варианта
m=[0 0]';
S=[2 0;0 2];
N=500;
X = mvnrnd(m,S,N)';
figure(3), plot(X(1,:),X(2,:),'.');
figure(3), axis equal
figure(3), axis([-7 7 -7 7])
% Рисуем точки для четвертого варианта
m=[0 0]';
S=[0.2 0;0 2];
N=500;
X = mvnrnd(m,S,N)';
figure(4), plot(X(1,:),X(2,:),'.');
figure(4), axis equal
figure(4), axis([-7 7 -7 7])
% Рисуем точки для пятого варианта
m=[0 0]';
S=[2 0;0 0.2];
N=500;
X = mvnrnd(m,S,N)';
figure(5), plot(X(1,:),X(2,:),'.');
figure(5), axis equal
figure(5), axis([-7 7 -7 7])
% Рисуем точки для шестого варианта
m=[0 0]';
S=[1 0.5;0.5 1];
N=500;
X = mvnrnd(m,S,N)';
figure(6), plot(X(1,:),X(2,:),'.');
figure(6), axis equal
figure(6), axis([-7 7 -7 7])
% Рисуем точки для седьмого варианта
m=[0 0]';
S=[.3 0.5;0.5 2];
N=500;
X = mvnrnd(m,S,N)';
```

```
figure(7), plot(X(1,:),X(2,:),'.');
figure(7), axis equal
figure(7), axis([-7 7 -7 7])
% Рисуем точки для восьмого варианта
m=[0 0]';
S=[.3 -0.5;-0.5 2];
N=500;
X = mvnrnd(m,S,N)';
figure(8), plot(X(1,:),X(2,:),'.');
figure(8), axis equal
figure(8), axis([-7 7 -7 7])
```

На рисунках 3/15 – 3.22 показано сгенерированное множество точек

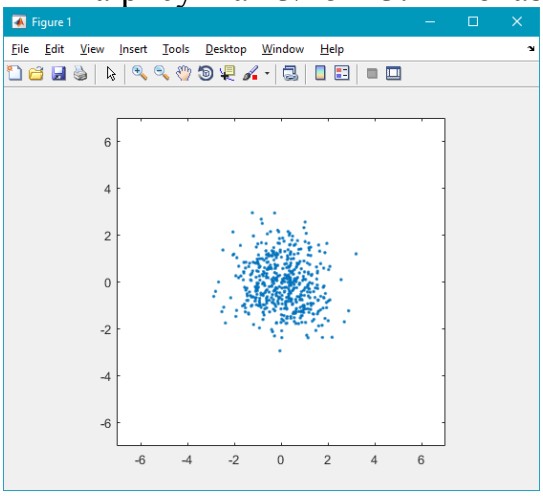


Рисунок 3.15 – Сгенерированное множество точек для случая $\sigma_1^2 = \sigma_2^2 = 1,\ \sigma_{12} = 0$

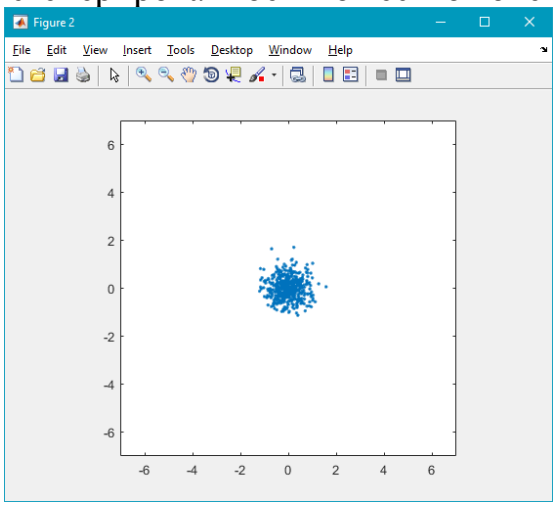


Рисунок 3.16 – Сгенерированное множество точек для случая $\sigma_1^2 = \sigma_2^2 = 0.2,\ \sigma_{12} = 0$

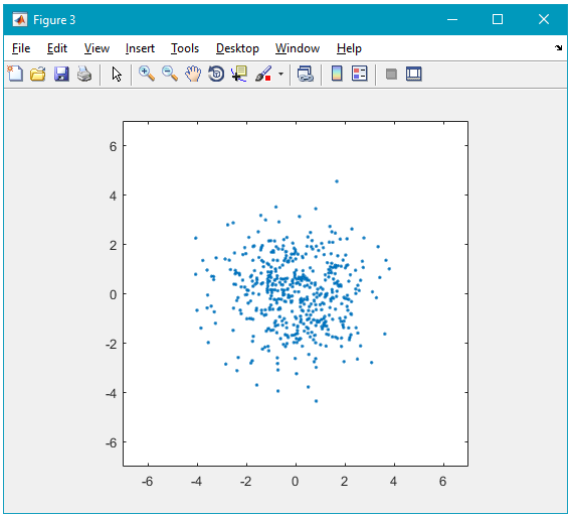


Рисунок 3.17 – Сгенерированное множество точек для случая $\sigma_1^2 = \sigma_2^2 = 2,\ \sigma_{12} = 0$

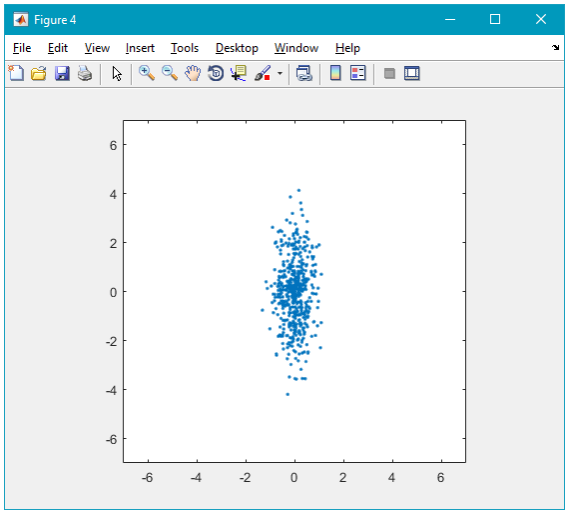


Рисунок 3.18 – Сгенерированное множество точек для случая $\sigma_1^2 = 0.2,\ \sigma_2^2 = 2,\ \sigma_{12} = 0$

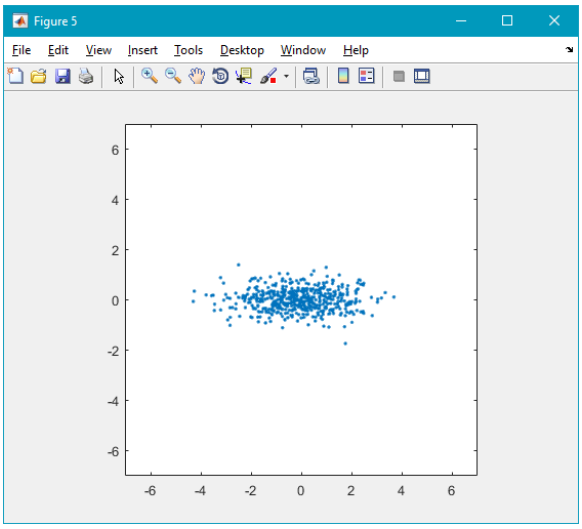


Рисунок 3.19 – Сгенерированное множество точек для случая $\sigma_1^2 = 2,\ \sigma_2^2 = 0.2,\ \sigma_{12} = 0$

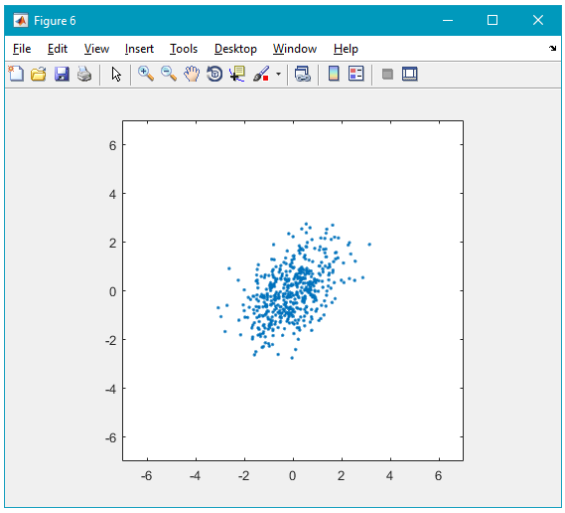


Рисунок 3.20– Сгенерированное множество точек для случая $\sigma_1^2 = \sigma_2^2 = 1,\ \sigma_{12} = 0.5$

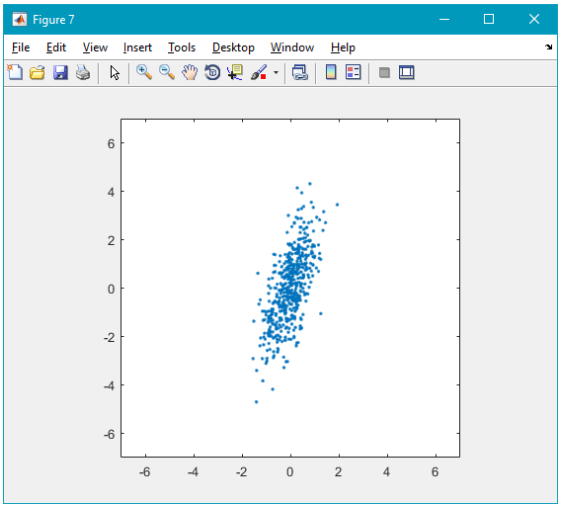


Рисунок 3.21– Сгенерированное множество точек для случая $\sigma_1^2 = 0.3,\ \sigma_2^2 = 2,\ \sigma_{12} = 0.5$

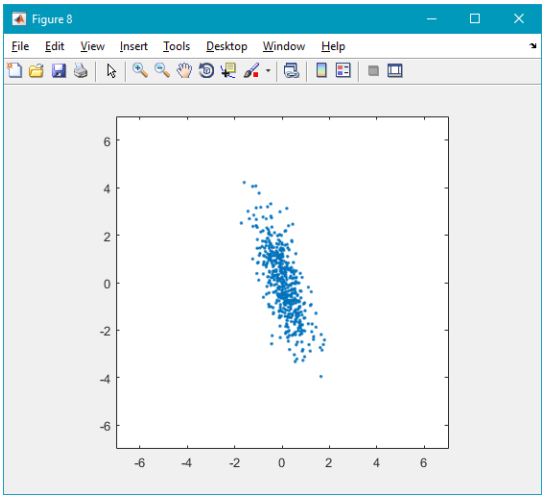


Рисунок 3.22– Сгенерированное множество точек для случая $\sigma_1^2 = 0.3,\ \sigma_2^2 = 2,\ \sigma_{12} = -0.5$

Из этих графиков можно сделать следующий вывод, если признаки некоррелированы, $S = diag(\sigma_1^2, \ldots, \sigma_n^2)$, то линии уровня плотности распределения имеют форму эллипсоидов с центром m и осями, параллельными линиям координат (рисунок 3.18 и 3.19). Если признаки имеют одинаковые дисперсии, то эллипсоиды являются сферами (рисунок 3.15 –3.17). Если признаки коррелированны, то матрица S не диагональная и линии уровня имеют форму эллипсоидов, оси которых повернуты относительно исходной системы координат (рисунок 3.20 – 3.22).

Итак, в коде байесовского классификатора мы определяли класс по формуле из теории вероятности: умножали вероятность соответствующего класса на плотность распределения, причем предполагали, что плотность – нормальная, а вероятности известны.

Но можно решать задачу классификации и путём вычисления

расстояний.

Допустим, у нас есть некая точка в виде *n*-мерного вектора, и есть выборка, которую тоже можно отразить в виде набора точек в *n*-мерном пространстве. Тогда, если мы сначала получим некие характеристики выборки, допустим, узнаем её центр-масс, то можно определить расстояние от точки до центра масс. Если выборок много, то какое расстояние меньше, тому классу и принадлежит точка.

Пример 6. Сначала используем формулу из школьного курса: евклидово расстояние.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2} = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}, \qquad (3.12)$$

где *p* и *q*, *n*-мерные вектора.

```
close('all');
clear;
% x – входной вектор
x=[0.1 0.5 0.1]';
% m1 и m2 играют роль оценки двух неизвестных множеств: их центры масс
m1=[0 0 0]';
m2=[0.5 0.5 0.5]';
m=[m1 m2];
% передача информации в функцию
z=euclidean_classifier(m,X)
```

```
>> close('all');
clear;
% x - входной вектор
X=[0.1 0.5 0.1]';
% m1 и m2 играют роль оценки двух н
масс
m1=[0 0 0]';
m2=[0.5 0.5 0.5]';
m=[m1 m2];
% передача информации в функцию
z=euclidean_classifier(m,X)

z =

     1
```

```
% функия в отдельном файле
function [z]=euclidean_classifier(m,X)
% вернёт индекс класса 1 или 2 к которому принадлежит точка X
% X – это матрица 1x3
% m – это матрица 3x2
% size() – возвращает координаты матрицы
[l,c]=size(m); % (l, c) = (3, 2)
[l,N]=size(X); % (l, N) = (3, 1)
for i=1:N % цикл по строкам X от 1 до 3
   for j=1:c % цикл по столбцам m от 1 до 2
      % высчитываем два евклидовых расстояния
      de(j)=sqrt((X(:,i)-m(:,j))'*(X(:,i)-m(:,j)));
   end
   % de(1) = 0.5196
   % de(2) = 0.5657
   % в num будет содержаться минимальное значение, т.е. de(1)
   % z(i) – будет содержать его индекс, т.е. 1
   [num,z(i)]=min(de);
end
```

Однако можно использовать и более сложную формулу расстояния: расстояние Махаланобиса, которое использует не только центр масс, но и матрицу корреляции.

Если мы оценили множества и нашли его среднее значение $m = [m_1, m_2, \ldots, m_n]^T$, а также матрицу ковариации S, то расстояние определится следующим образом от точки $x = [x_1, x_2, \ldots, x_n]^T$ до m.

$$D_M(x) = \sqrt{(x - m)^T S^{-1} (x - m)}. \quad (3.13)$$

Тогда точка будет принадлежать тому классу для которого выполняется условие:

$$\sqrt{(x - m_i)^T S^{-1} (x - m_i)} < \sqrt{(x - m_j)^T S^{-1} (x - m_j)}, \forall j \neq i, \quad (3.14)$$

т.е. для которого расстояние Махаланобиса минимально.

Пример 7. Напишем код для соответствующей классификации.

```
close('all');
clear;
x=[0.1 0.5 0.1]';
m1=[0 0 0]';
m2=[0.5 0.5 0.5]';
m=[m1 m2];
% для двух множеств одна и таже матрица ковариации S
S=[0.8 0.01 0.01;
0.01 0.2 0.01;
0.01 0.01 0.2];
z=mahalanobis_classifier(m,S,x)
```

```
>> close('all');
clear;
x=[0.1 0.5 0.1]';
m1=[0 0 0]';
m2=[0.5 0.5 0.5]';
m=[m1 m2];
% для двух множеств одна и таже :
S=[0.8 0.01 0.01;
   0.01 0.2 0.01;
   0.01 0.01 0.2];
z=mahalanobis_classifier(m,S,x)

z =

     2
```

```
% функция для расстояния Махаланобиса
function z=mahalanobis_classifier(m,S,X)
[l,c]=size(m);
[l,N]=size(X);
for i=1:N
   for j=1:c
      dm(j)=sqrt((X(:,i)-m(:,j))'*inv(S)*(X(:,i)-m(:,j)));
   end
% dm(1) = 1.1334
% dm(2) = 0.9918
   [num,z(i)]=min(dm);
end
% z = 2
```

Мы видим, что в этом случае точка принадлежит классу 2.

Получается противоречие: евклидово расстояние показывает принадлежность к классу 1, а расстояние Махаланобиса – наоборот. В таком случае нужно принять ответ классификатора Махаланобиса, т.к. за счёт матрицы ковариации расстояние Махаланобиса учитывает ещё и форму распределения точек вокруг центра масс. По сути расстояние Махаланобиса – это просто расстояние между заданной точкой и центром масс, делённое на ширину эллипсоида в направлении заданной точки.

Рассмотрим далее принцип максимума правдоподобия, который

составляет основу параметрического подхода. Пусть задано множество объектов $X^m = x_1,...,x_m$, выбранных независимо друг от друга из вероятностного распределения с плотностью φ(x;θ). Функцией правдоподобия называется совместная плотность распределения всех объектов выборки:

$$p(X^m;\theta) = p(x_1,...,x_m;\theta) = \prod_{i=1}^{m} \varphi(x_i;\theta) \quad (3.15)$$

Значение параметра θ, при котором выборка максимально правдоподобна, то есть функция $p(X^m;\theta)$ принимает максимальное значение, называется оценкой максимума правдоподобия.

В случае гауссовой плотности с параметрами $\theta = (m, S)$ задача максимизации правдоподобия имеет аналитическое решение:

$$m_{ML} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad (3.16)$$

$$S_{ML} = \frac{1}{N}\sum_{i=1}^{N} (x_i - m_{ML})(x_i - m_{ML})^T. \quad (3.17)$$

Пример 8. Рассмотрим на примере как работает алгоритм максимума правдоподобия. Сгенерируем 50 точек, имеющих по две координаты каждая, используя гауссово распределение с центром m = $[2, -2]^T$, $S = \begin{bmatrix} 0.9 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}$. Потом оценим параметры этого распределения по формулам (3.16) и (3.17), сравним результаты.

```
close('all');
clear;
% Генерируем 50 точек с заданным распределением гаусса
randn('seed',0);
m = [2 -2]; S = [0.9 0.2; 0.2 .3];
X = mvnrnd(m,S,50)';
% Оцениваем параметры распределения по выборке
[m_hat,
S_hat]=Gaussian_ML_estimate(X)
```

```
>> close('all');
clear;
% Генерируем 50 точек с заданным распре
randn('seed',0);
m = [2 -2]; S = [0.9 0.2; 0.2 .3];
X = mvnrnd(m,S,50)';
% Оцениваем параметры распределения по
[m_hat, S_hat]=Gaussian_ML_estimate(X)

m_hat =

    2.0495
   -1.9418


S_hat =

    0.8082    0.0885
    0.0885    0.2298
```

Функцию можно создать в отдельном файле Matlab.

```
function [m_hat,S_hat]=Gaussian_ML_estimate(X)
% Входной параметр:
%   X:     lxN матрица, чьи колонки есть наши данные.
% Выходной аргумент:
%   m_hat:  l-размерная оценка среднего вектора распределения.
%   S_hat:  lxl оценка ковариационной матрицы распределения.
 [l,N]=size(X); % l = 2, N = 50
m_hat=(1/N)*sum(X')'; % получаем среднеарифметический центр масс
% создаём и инициализируем 0 квадратную матрицу 2x2
```

```
S_hat=zeros(l);
for k=1:N
   S_hat=S_hat+(X(:,k)-m_hat)*(X(:,k)-m_hat)';
end
% Высчитываем среднее
S_hat=(1/N)*S_hat;
```

По результатам получился вектор m_hat = $[2.0495, -1.9418]^T$, $S\_hat = \begin{bmatrix} 0.8082 & 0.0885 \\ 0.0885 & 0.2298 \end{bmatrix}$.

Как видно, получившиеся оценки близки к оригиналу, однако, нужно учесть, что они проводились на выборке, состоящей из 50 элементов, что очень мало, поэтому эти оценки не надёжны. Для увеличения надёжности оценок нужно увеличить выборку.

Пример 9. Теперь рассмотрим последний пример, в котором обобщим весь предыдущий материал. Сгенерируем два множества X и X_1 каждое содержит 999 трехмерных точек, X – обучающее множество, X_1 – тестовое. Каждое множество включает три равновероятных класса: w_1, w_2, w_3; классы создаются с помощью распределения Гаусса со средними $m_1 = [0, 0, 0]^T$, $m_2 = [1, 2, 2]^T$, $m_3 = [3, 3, 4]^T$ и матрицами ковариации

$$S_1 = S_2 = S_3 = \begin{bmatrix} 0.8 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.8 \end{bmatrix} = \sigma^2 I$$

Используя X с помощью принципа максимального правдоподобия оценим такие параметры как среднее m и матрицы ковариации S_1, S_2, S_3, т.к. у нас 10^3 точек, то эти оценки будут надёжными. Далее, используя эти оценки проведём классификацию с помощью расстояния Махаланобиса, Евклида, а также байесовским классификатором. Для каждого способа вычислим ошибку вероятности и сравним результаты.

Сначала сгенерируем обучающее множество X.

```
close('all');
clear;
% Матрица m 3x3
m=[0 0 0; 1 2 2; 3 3 4]';
% Создаём матрицу S1 3x3 (единичную матрицу умножаем на 0.8)
S1=0.8*eye(3);
% Как переменная m содержит три вектора-центра масс, так и тут
% трехмерный массив S содержит три одинаковых матрицы ковариации
S(:,:,1)=S1;S(:,:,2)=S1;S(:,:,3)=S1;
% вероятность появления каждого из трех классов
P=[1/3 1/3 1/3]';
% количество элементов в обучающем множестве
N=1000;
% устанавливаем датчик случайных чисел с параметрами seed и 0.
randn('seed',0);
% Генерируем множество X[3;999] – сами точки, y[1, 999] – их метки
% от 1 до 333 – метка 1, от 334 до 666 – метка 2, от 667 до 999 – метка 3.
```

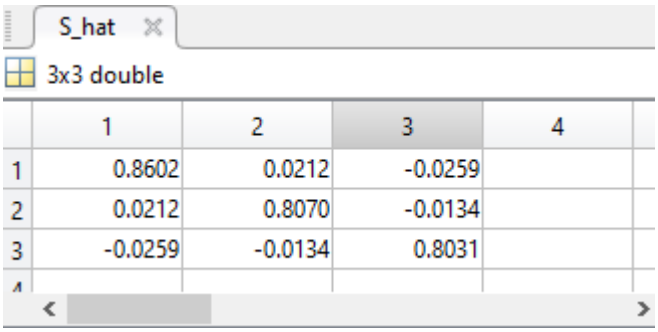
```
[X,y]=generate_gauss_classes(m,S,P,N);

% Функцию пишем в отдельном файле
function [X,y]=generate_gauss_classes(m,S,P,N)
[l,c]=size(m); % (l, c) = (3, 3)
X=[]; % Инициализация множеств
y=[];
for j=1:c % цикл от 1 до 3
% Генерируем трехмерные точки со средним m(:, j), где ":" – для всех строк,
% матрицей ковариации S(:, :, j) и количеством 1/3*1000, ограниченным снизу
  t=mvnrnd(m(:,j),S(:,:,j),fix(P(j)*N))';
  % сгенерированные точки присваиваем вектору X причем так, чтобы
  % они добавлялись в конец уже существующего вектора X. Сгенерировали
  % точки для j = 1, добавили их к пустому вектору, затем сгенерировали
  % точки для j = 2, добавили их в конец вектора, который уже содержит
  % точки для j = 1 и т.д.
  X=[X t]; % генерируем метки
  y=[y ones(1,fix(P(j)*N))*j];
end
```

Далее аналогично сгенерируем множество X_1, но уже с другими параметрами для **randn()**.

```
% Генерируем тестовую выборку X1
randn('seed',100);
[X1,y1]=generate_gauss_classes(m,S,P,N);
```

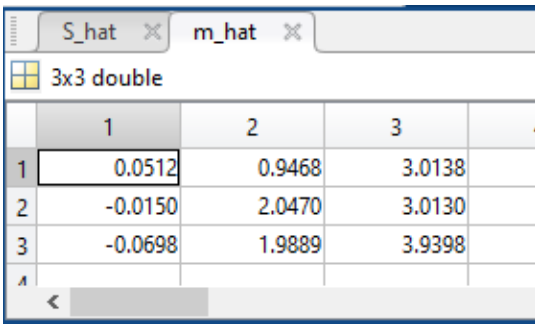
Теперь по принципу максимума правдоподобия вычислим оценки для распределения, которое использовалось при создании множества X.

```
% извлекаем из X только те точки, которые имеют метку 1
class1_data=X(:,find(y==1));
% вычисляем центр масс и ковариационную матрицу
% функция для вычисления была разобрана ранее
[m1_hat, S1_hat]=Gaussian_ML_estimate(class1_data);
% Аналогично
class2_data=X(:,find(y==2));
[m2_hat, S2_hat]=Gaussian_ML_estimate(class2_data);
% Аналогично
class3_data=X(:,find(y==3));
[m3_hat, S3_hat]=Gaussian_ML_estimate(class3_data);
% Получаем общую оценку единой ковариационной матрицы,
% как среднеарифметическое трех уже вычисленных матриц
S_hat=(1/3)*(S1_hat+S2_hat+S3_hat);
m_hat=[m1_hat m2_hat m3_hat];
```


S_hat

3x3 double

	1	2	3	4
1	0.8602	0.0212	-0.0259	
2	0.0212	0.8070	-0.0134	
3	-0.0259	-0.0134	0.8031	


S_hat | m_hat

3x3 double

	1	2	3
1	0.0512	0.9468	3.0138
2	-0.0150	2.0470	3.0130
3	-0.0698	1.9889	3.9398

В итоге получаем:

$$S\_hat = \begin{bmatrix} 0.8602 & 0.0212 & -0.0259 \\ 0.0212 & 0.8070 & -0.0134 \\ -0.0259 & -0.0134 & 0.8031 \end{bmatrix} \quad \text{и} \quad m\_hat = \begin{bmatrix} 0.0512 & 0.9468 & 3.0138 \\ -0.0150 & 2.0470 & 3.0130 \\ -0.0698 & 1.9889 & 3.9398 \end{bmatrix}.$$

Видно, что получившиеся оценки близки к тем, которые мы задавали изначально. В реальности S и m из условия задачи нам нужны были только для того, чтобы создать выборки. Предполагается, что выборки X и X_1 мы получаем откуда-то со стороны и делаем предположение, что они сформированы посредством нормального распределения. Такое предположение мы можем сделать, т.к. рисунки 3.15 – 3.22 демонстрируют, что гауссово распределение может моделировать широкий класс данных.

Теперь, зная характеристики выборки, точнее трёх классов, входящих в неё, можно провести классификацию точек из тестового множества X_1.

```
% высчитываем расстояние от точки X1[i], i=1..999 до точки-центра
% соответствующего класса, z_euclidean будет содержать 999 индексов классов
% с самым минимальным расстоянием
z_euclidean=euclidean_classifier(m_hat,X1);
% Аналогично
z_mahalanobis=mahalanobis_classifier(m_hat,S_hat,X1);
% Используем байесовский классификатор, однако с математической
% точки зрения, чтобы он работал корректно, мы должны подавать ему
% не вычисленные оценки множества X, а данные в условии, т.е. точные
z_bayesian=bayes_classifier(m,S,P,X1);
% Находим процент несовпадений по классам
err_euclidean = (1-length(find(y1==z_euclidean))/length(y1))
err_mahalanobis = (1-length(find(y1==z_mahalanobis))/length(y1))
err_bayesian = (1-length(find(y1==z_bayesian))/length(y1))

% Функция для классификатора байеса
function [z]=bayes_classifier(m,S,P,X)
[l,c]=size(m);
[l,N]=size(X);
for i=1:N % цикл от 1 до 999
   for j=1:c % цикл от 1 до 3
      t(j)=P(j)*comp_gauss_dens_val(m(:,j),S(:,:,j),X(:,i));
   end
   [num,z(i)]=max(t);
```

end

Получается, что во всех трех случаях ошибки похожи: 7.61%, 7.71%, 7.61%, рисунок 3.23.

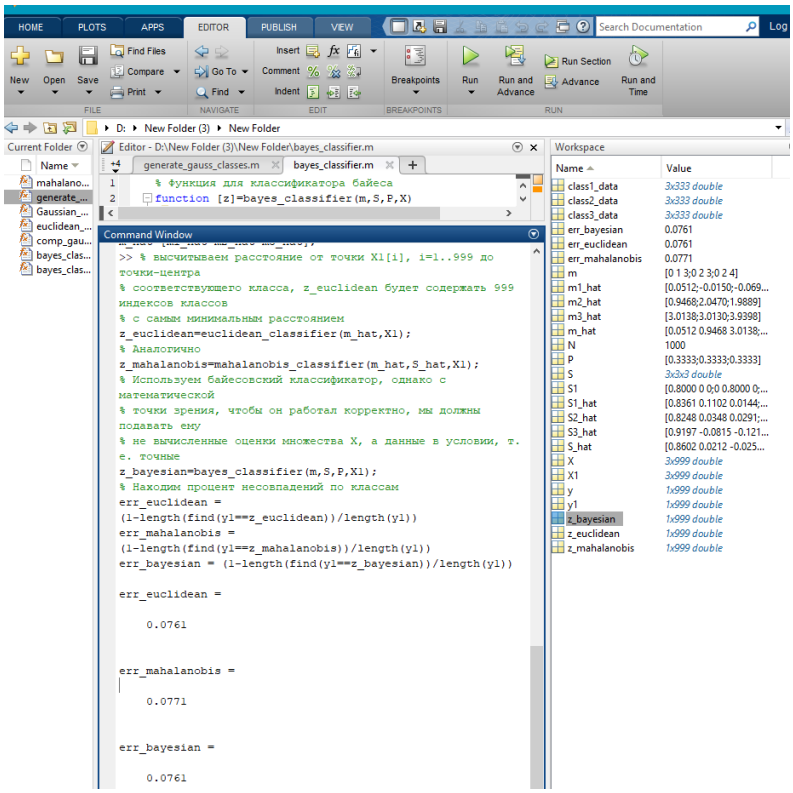


Рисунок 3.23 – Рабочее окно Matlab с результатами классификации с помощью расстояния Махаланобиса, Евклида и байесовским классификатором

Это не случайно, если выполняются 4 следующих условия, то все три метода классификации эквивалентны:

1. Все классы равновероятны.
2. Данные во всех классах имеют гауссовы распределения.
3. Матрицы ковариации одни и те же.
4. Ковариационные матрицы не просто равны, но и диагональные, причем все элементы на главной диагонали равны

Задание 3.1. Проверьте работу программного кода из второго примера для классификации точек плоскости на множества точек «вне» и «внутри» эллипса, для чего внесите следующие изменения в код и проанализируйте полученные результаты:

1) в качестве третьего параметра функции newp укажите другие модели персептрона (сигмоидальные функции): 'tansig' — гиперболический тангенс; 'logsig' — логистический сигмоид; 'purelin' — линейный сигмоид;

2) измените максимальное число итераций для проведения обучения сети;

3) проверьте корректность работы построенной нейронной сети, для чего проведите классификацию нескольких тестовых точек плоскости; для наглядности выполните построение этих точек на той же плоскости;

координаты тестовых точек задайте самостоятельно;

4) измените параметры исходного эллипса (коэффициенты кривой).

Задание 3.2. Рассмотрите, как в примере 3 параболу, заданную в виде общего уравнения кривой второго порядка. Сгенерируйте случайное множество на плоскости (100 точек). Используя нелинейное отображение $\psi(x,y)$ в 5-мерное пространство и однослойный персептрон в 5-мерном пространстве, разделите точки множества на два класса – точки, лежащие «вне» и «внутри» параболы.

Задание 3.2. Вычислите как в примере 4 значения p_1 и p_2, но для следующих двух случаев: ($P(w_1)$=1/6, $P(w_2)$=5/6), ($P(w_1)$=5/6, $P(w_2)$=1/6). Дайте объяснения зависимости результатов классификации от априорных вероятностей.

Задание 3.2. Повторите пример 8, но для N = 500 и N = 5000 точек. Прокомментируйте результат.

В таблице 3.1 приведены индивидуальные задания.

Таблица 3.1 – Индивидуальные варианты

1	Повторите пример 9, но где $S_1 = S_2 = S_3 = \begin{bmatrix} 0.8 & 0.2 & 0.1 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.2 & 0.8 \end{bmatrix} \neq \sigma^2 I$. Напишите вывод по получившемуся ответу.
2	Повторите пример 9, но где для генерации X и X_1 будут использованы следующие вероятности классов P_1 = 1/2, P_2 = P_3 = 1/4. Для этого случая байесовский классификатор должен показать лучший результат. Почему?
3	Повторите пример 9, но где $P(w1)$ = P($w2$) = P($w3$) = 1/3 и $S_1 = \begin{bmatrix} 0.8 & 0.2 & 0.1 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.2 & 0.8 \end{bmatrix}$, $S_2 = \begin{bmatrix} 0.6 & 0.01 & 0.01 \\ 0.01 & 0.8 & 0.01 \\ 0.01 & 0.01 & 0.6 \end{bmatrix}$, $S_3 = \begin{bmatrix} 0.6 & 0.1 & 0.1 \\ 0.1 & 0.6 & 0.1 \\ 0.1 & 0.1 & 0.6 \end{bmatrix}$. Объясните результат.
4	Повторите пример 9, но где $S_1 = S_2 = S_3 = \begin{bmatrix} 0.8 & 0.3 & 0.1 \\ 0.3 & 0.8 & 0.3 \\ 0.1 & 0.3 & 0.8 \end{bmatrix} \neq \sigma^2 I$. Напишите вывод по получившемуся ответу.
5	Повторите пример 9, но где для генерации X и X_1 будут использованы следующие вероятности классов P_1 = 0.8, P_2 = 0.15, P_3 = 0.05. Прокомментируйте результат.
6	Повторите пример 9, но где $P(w1)$ = $P(w2)$ = $P(w3)$ = 1/3 и $S_1 = \begin{bmatrix} 0.9 & 0.2 & 0.1 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.2 & 0.9 \end{bmatrix}$

	, $S_2 = \begin{bmatrix} 0.6 & 0.01 & 0.01 \\ 0.01 & 0.8 & 0.01 \\ 0.01 & 0.01 & 0.6 \end{bmatrix}$, $S_3 = \begin{bmatrix} 0.6 & 0.2 & 0.1 \\ 0.1 & 0.6 & 0.1 \\ 0.1 & 0.2 & 0.6 \end{bmatrix}$. Объясните результат.
7	Повторите пример 9, но где $S_1 = S_2 = S_3 = \begin{bmatrix} 0.7 & 0.2 & 0.4 \\ 0.2 & 0.7 & 0.2 \\ 0.4 & 0.2 & 0.7 \end{bmatrix} \neq \sigma^2 I$. Напишите вывод по получившемуся ответу.
8	Повторите пример 9, но где для генерации X и X_1 будут использованы следующие вероятности классов $P_1 = 0.6$, $P_2 = P_3 = 0.2$ Для этого случая байесовский классификатор должен показать лучший результат. Почему?
9	Повторите пример 9, но где $P(w1) = P(w2) = 0.2$, $P(w3) = 0.6$ и $S_1 = \begin{bmatrix} 0.8 & 0.2 & 0.1 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.2 & 0.8 \end{bmatrix}$, $S_2 = \begin{bmatrix} 0.6 & 0.01 & 0.01 \\ 0.01 & 0.8 & 0.01 \\ 0.01 & 0.01 & 0.6 \end{bmatrix}$, $S_3 = \begin{bmatrix} 0.6 & 0.1 & 0.1 \\ 0.1 & 0.6 & 0.1 \\ 0.1 & 0.1 & 0.6 \end{bmatrix}$. Объясните результат.
10	Повторите пример 9, но где для генерации X и X_1 будут использованы следующие вероятности классов $P_1 = 0.4$, $P_2 = P_3 = 0.3$. Для этого случая байесовский классификатор должен показать лучший результат. Почему?

Содержание отчета и его форма

Подготовьте отчет, в котором приведите технологию выполнения заданий.

Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) формулировку задания и технологию его выполнения;
4) ответы на контрольные вопросы;
5) приложение – файлы выполненных заданий.

Вопросы для защиты работы

1. Что такое матрица ковариации?
2. Что такое дисперсия и математическое ожидание?
3. Чем отличается классификация на основе расстояния Махаланобиса от классификации на основе расстояния Евклида?
4. Что такое байесовский классификатор?
5. Что такое принцип максимума правдоподобия?

ЛАБОРАТОРНАЯ РАБОТА № 4
Классификаторы, основанные на оценки функции оптимизации

Цель и содержание: изучить основные приемы работы с классификаторами, основанными на оценки функции оптимизации.

Организационная форма занятий: решение проблемных задач, разбор конкретных ситуаций.

Вопросы для обсуждения на лабораторном занятии: принцип действия и алгоритм работы персептрона, алгоритм минимизации среднеквадратической ошибки (LMS). построение классификаторов и их использование в MATLAB.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Техника получения оптимального байесовского классификатора основывается на оценке функции распределения Гаусса для обучающих данных. Однако, это сложная задача особенно для многомерных данных.

Альтернативное решение заключается в отделении классов в многомерном пространстве с помощью разделяющих гиперплоскостей. Разделяющая гиперплоскость может быть записана в виде

$$w^T x + w_0 = 0 \quad (4.1)$$

Также (4.1) можно переписать в виде

$$w'^T x' \equiv [\mathrm{w}^T, \mathrm{w}_0] \begin{bmatrix} x \\ 1 \end{bmatrix} = 0 \quad (4.2)$$

где $\boldsymbol{w}$ – вектор настраиваемых параметров, w_0 – свободный член, $\boldsymbol{x}$ – входной вектор.

С помощью таких гиперплоскостей можно отделить линейно отделимые классы.

Алгоритм адаптации вектора весовых коэффициентов элементарного персептрона можно сформулировать следующим образом.

Если n-ый элемент $x(n)$ обучающего множества корректно классифицирован с помощью весовых коэффициентов $w(n)$, вычисленных на n-ом шаге алгоритма, то вектор весов не корректируется, т.е. действует следующее правило:

$$\begin{aligned} w(n+1) &= w(n), \text{ если } w^T x(n) > 0 \text{ и } x(n) \in C_1 \\ w(n+1) &= w(n), \text{ если } w^T x(n) \le 0 \text{ и } x(n) \in C_2 \end{aligned} \quad (4.3)$$

В противном случае вектор весов персептрона подвергается коррекции в соответствии со следующим правилом:

$$\begin{aligned} w(n+1) &= w(n) - \eta(n)\, x(n), \text{ если } w^T(n) x(n) > 0 \text{ и } x(n) \in C_2 \\ w(n+1) &= w(n) + \eta(n)\, x(n), \text{ если } w^T(n) x(n) \le 0 \text{ и } x(n) \in C_1 \end{aligned}, \quad (4.4)$$

где C_1 и C_2 – линейно разделимые классы, а $\eta(n)$ – параметр скорости обучения.

Создадим функцию, которая будет обучать наш линейный нейрон (настраиваемую гиперплоскость). Сигнатура функции будет следующей: **[w, iter, mis_clas] = perce(X, y, w_ini. rho)**, где *X* – это матрица размером (l+1)xN (l – размер входного вектора, N – количество паттернов для обучения), y – вектор меток для класса C_1 (+1) и C_2 (-1), w_ini – вектор начальных параметров, которые нужно настроить в процессе обучения так, чтобы гиперплоскость отделяла C_1 от C_2, rho – η=const, т.е. скорость обучения, w – вектор, вычисленный алгоритмом, iter – количество итераций за которые был вычислен w, mis_clas – количество неправильно классифицированных векторов.

Рассмотрим примеры.

Пример 1. Необходимо сгенерировать 4 множества X_i, каждое – это набор точек в двумерном пространстве. Каждая точка из X_i имеет метку -1, точки случайно разбросаны в квадрате [3, 5]x[3, 5], [2, 4]x[2, 4], [0, 2]x[2, 4], [1, 3]x[1, 3]. Точки, распложенные в квадрате [0, 2]x[0,2] имеют метку класса +1. Требуется визуально отобразить классы, обучить персептрон для η=0.01 и η=0.05, и начальном векторе параметров $[1, 1, -0.5]^T$.

Листинг функции, реализующий персептрон, приведён ниже.

```
function [w,iter,mis_clas]=perce(X,y,w_ini,rho)

[l,N]=size(X);
max_iter=20000; % Максимальное количество итераций,
% которые будет работать персептрон, после этого будет
% считаться, что решение не найдено.

w=w_ini;       % Инициализируем вектор параметров
iter=0;        % Счётчик итераций

mis_clas=N;    % Инициализируем количество неправильно
% классифицированных паттернов

% цикл while по двум условиям
while(mis_clas>0)&&(iter<max_iter)
   iter=iter+1;
   mis_clas=0;

   gradi=zeros(l,1); % Вычисление корректирующего компонента для вектора
   for i=1:N
      if((X(:,i)'*w)*y(i)<0)
         mis_clas=mis_clas+1;
         gradi=gradi+rho*(-y(i)*X(:,i));
      end
   end

   if(iter==1)
      fprintf('\n First Iteration: # Misclassified points = %g \n',mis_clas);
   end
```

```
   w=w-rho*gradi; % Обновление вектора параметров
end
```

Листинг основного кода приведён ниже.

```
close('all');
clear
rand('seed',0);
% Генерируем класс Xi
N=[100 100]; % 100 векторов для каждого класса
l=2; % Размерность входа каждого вектора
 % для генерирования паттернов из X2, X3, X4 нужно вместо
% нижней строчки подставить одну из закомментированных строк
x=[3 3]';
% x=[2 2]'; для X2
% x=[0 2]'; для X3
% x=[1 1]'; для X4
% получаем 200 точек для Xi и для точек квадрата [0, 2]x[0, 2]
X1=[2*rand(l,N(1)) 2*rand(l,N(2))+x*ones(1,N(2))];
X1=[X1; ones(1,sum(N))];
y1=[-ones(1,N(1)) ones(1,N(2))]; % получаем метки
1. Выводим множество Xi
figure(1), plot(X1(1,y1==1),X1(2,y1==1),'bo',...
X1(1,y1==-1),X1(2,y1==-1),'r.')
figure(1), axis equal
% 2. Запускаем алгоритм обучения персептрна для эта=0.01
rho=0.01; % Скорость обучения
w_ini=[1 1 -0.5]'; % начальные веса
[w,iter,mis_clas]=perce(X1,y1,w_ini,rho)
```

Имеем 4 варианта X_i, i = 1..4, которые нужно отделить с помощью персептрона (рисунок 4.1 – 4.4).

Видно из рисунка X_4 (рисунок 4.4) классы линейно не разделимы.

Результаты обучения приведены в таблице 4.1, 4.2. В таблице 4.1 представлено количество итераций, за которое персептрон находит решение в зависимости от η, а в таблице 4.2 количество неправильно распознанных паттернов.

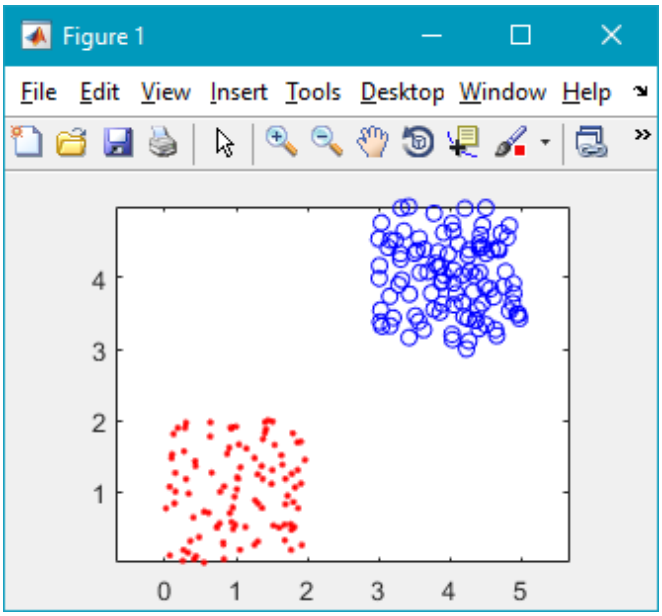


Рисунок 4.1 – Два класса для X_1

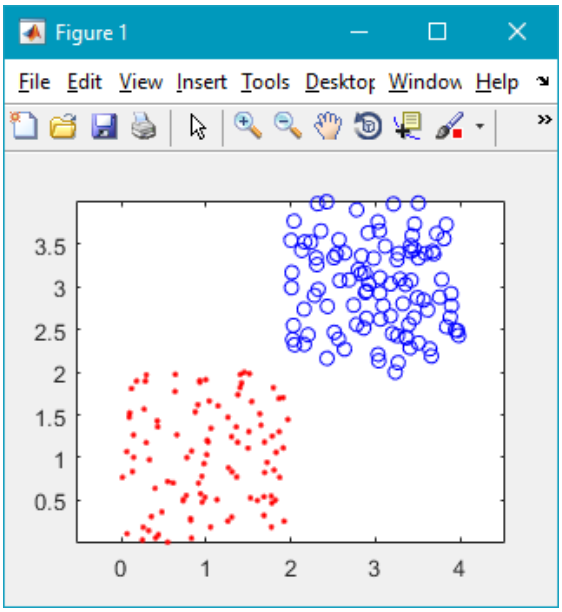


Рисунок 4.2 – Два класса для X_2

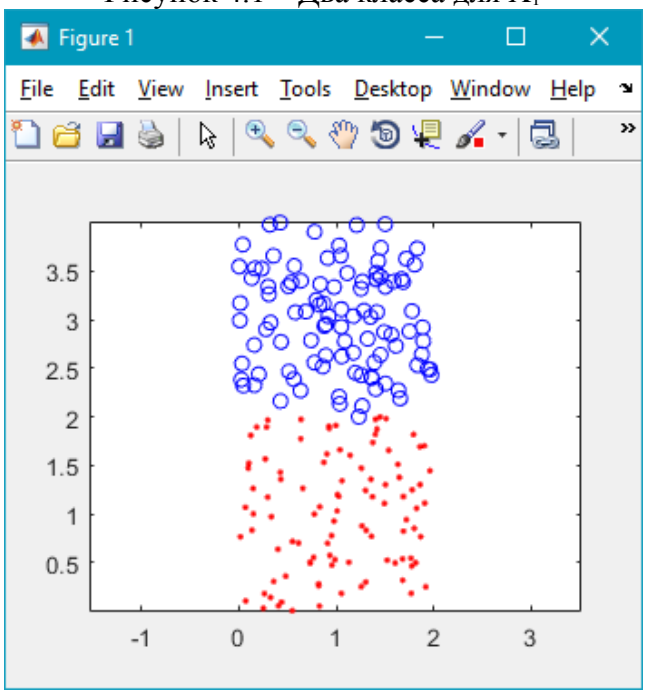


Рисунок 4.3 – Два класса для X_3

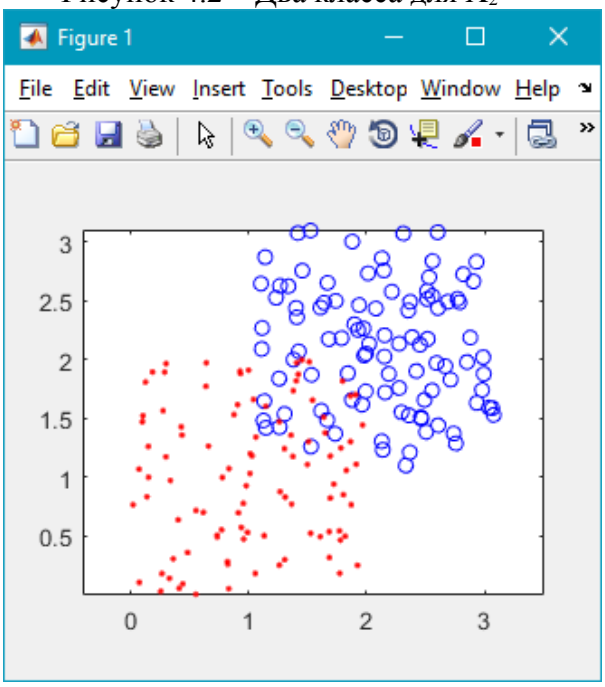


Рисунок 4.4 – Два класса для X_4

Таблица 4.1 – Количество итераций, за которые персептрон находит решение в зависимости от η

	X_1	X_2	X_3	X_4
η = 0.01	134	134	5441	Оптимального решения не найдено
η = 0.05	5	5	252	Оптимального решения не найдено

Таблица 4.2 – Количество неправильно распознанных паттернов

	X_1	X_2	X_3	X_4
η = 0.01, η = 0.05	0	0	0	25

При увеличении η увеличивается скорость обучения персептрона. В

последнем случае оптимального решения персептрон не находит, поэтому проведённая прямая отделяет не все паттерны класса.

Функция **perce(X,y,w_ini,rho)** запрограммирована для пакетного режима, т.е. вектор **w** корректируется один раз после вычисления поправок для каждого примера.

Пример 2. Обучение персептрона в онлайн-режиме. Ниже приведён листинг функции, где персептрон обучается в онлайн-режиме, где корректировки вычисляются для каждого примера, но далее, они не накапливаются, а сразу применяются к вектору весов.

```
function [w,iter,mis_clas]=perce_online(X,y,w_ini,rho)

[l,N]=size(X);
max_iter=10000000; % максимальное количество итераций
% требуется, если классы расположены близко друг к другу

w=w_ini;
iter=0;

mis_clas=N;
while(mis_clas>0)&&(iter<max_iter)
   mis_clas=0;

   for i=1:N
      if((X(:,i)'*w)*y(i)<0)
         mis_clas=mis_clas+1;
         w=w+rho*y(i)*X(:,i);  % Обновляем вектор весов
      end
      iter=iter+1;
   end

   if(iter==1)
      mis_clas
   end
end
```

Вызвать эту функцию можно с помощью следующей сигнатуры: **[w,iter,mis_clas]=perce_online(X1,y1,w_ini,rho).**

Если повторить предыдущий эксперимент, используя выше указанную функцию, при применении которой персептрон обучается в онлайн-режиме, то получим следующие результаты, таблица 4.3, 4.4.

Таблица 4.3 – Количество итераций, за которые персептрон находит решение в зависимости от η

	X_1	X_2	X_3	X_4
η = 0.01	600	600	6589400	Оптимального решения не найдено
η = 0.05	400	400	7729200	Оптимального решения не найдено

Таблица 2.4 – Количество неправильно распознанных паттернов

	X_1	X_2	X_3	X_4
η = 0.01, η = 0.05	0	0	0	6

Требуется значительно больше итераций, онлайн-обучение хуже сходится, зато в практическом плане более экономное, т.к. не требует специальных структур для хранения и накопления поправок.

Аппаратура и материалы. 64-разрядный (x64) персональный компьютер, процессор с тактовой частотой 1 ГГц и выше, оперативная память 1 Гб и выше, свободное дисковое пространство не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система Windows 7 и выше, Matlab (R2013) и выше.

Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

Выполните предложенные задания, предварительно рассмотрев приведенные в теоретической части примеры.

Задание 4.1 Создать два частично пересекающихся класса точек (100 точек для каждого класса) как показано в таблице 4.5.

Задание 4.2 Разделить их насколько это возможно с помощью персептрона. Тип обучения персептрона онлайн или пакетный указан в таблице.

Задание 4.3 Нарисовать найденное решение.

Таблица 4.5 – Индивидуальные варианты

№ Варианта	Задание	№ Варианта	Задание
1	$\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}$ Онлайн обучение.	6	$\begin{pmatrix} A & 0 \\ B & 0 \end{pmatrix}$ Пакетное обучение.
2	$\begin{pmatrix} A & 0 \\ B & 0 \end{pmatrix}$ Пакетное обучение.	7	$\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}$ Онлайн обучение.
3	$\begin{pmatrix} 0 & 0 \\ A & B \end{pmatrix}$ Пакетное обучение.	8	$\begin{pmatrix} A & 0 \\ B & 0 \end{pmatrix}$ Онлайн обучение.
4	$\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}$ Онлайн обучение.	9	$\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}$ Пакетное обучение.
5	$\begin{pmatrix} 0 & 0 \\ A & B \end{pmatrix}$ Пакетное обучение.	10	$\begin{pmatrix} 0 & 0 \\ A & B \end{pmatrix}$ Онлайн обучение.

Содержание отчета и его форма

Подготовьте отчет, в котором приведите технологию выполнения заданий.

Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) формулировку задания и технологию его выполнения;
4) ответы на контрольные вопросы;
5) приложение – файлы выполненных заданий.

Вопросы для защиты работы

1. Что такое персептрон?
2. Чем персептрон отличается от сети прямого распространения?
3. Какие задачи можно решать с помощью персептрона?

ЛАБОРАТОРНАЯ РАБОТА 5
Работа с нейронными сетями в MatLab

Цель и содержание: создать и обучить несколько нейронных сетей, решающих задачи линейного и нелинейного разделения классов, используя встроенный язык программирования Matlab

Организационная форма занятий: решение проблемных задач, разбор конкретных ситуаций

Вопросы для обсуждения на лабораторном занятии: освоить базовые команды для создания и обучения простых сетей в Matlab; научиться создавать и обучать сети для линейного и нелинейного разделения классов; научиться строить графики, отражающие результаты работы сетей.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Самый гибкий способ по работе с ИНС в Matlab – это использовать встроенный язык программирования для создания и обучения сетей. В данной лабораторной работе на примере линейного разделения классов и задачи XOR будет рассмотрена техника создания и обучения разных нейронных сетей.

Код Matlab будем выделять жирным шрифтом. Комментарий к коду записывается после символа «%» и идёт до конца строчки. При желании, можно получить более подробную справку о любой функции, установив курсор на нужной и нажав клавишу «F1».

Прежде, чем программировать персептрон, рассмотрим пример построения поверхности отклика для обычного нелинейного нейрона с функцией активации – гиперболический тангенс.

Пример 1. Построение поверхности отклика для нелинейного нейрона с функцией активации – гиперболический тангенс

```
% Готовим Matlab к работе
close all, clear all, clc, format compact;
% Задаём веса нейрона
w = [4 -2];
% Задаём смещение
b = -3;
% Задаём функцию, которую хотим построить, гипербол. тангенс
func = 'tansig';
% Задаём входной вектор
p = [2 3];
% Вычисляем взвешенную сумму входа и весов
activation_potential = p*w'+b;
% Вычисляем выход нелинейной части нейрона
neuron_output = feval(func, activation_potential);
% Задаём входной вектор для диапазона от -10 до +10 с шагом 0.25
[p1, p2] = meshgrid(-10:.25:10);
% Вычисляем вектор выхода нейрона
```

```
z = feval(func, [p1(:) p2(:)]*w'+b);
% Пересчёт вещественных чисел в воксели для 3D-графика
z = reshape(z, length(p1), length(p2));
% Строим график
plot3(p1, p2, z);
% Включаем сетку
grid on;
% Подписываем оси графика
xlabel('Input 1');
ylabel('Input 2');
zlabel('Neuron output');
```

Разберём код. Команда **close all** закрывает все вспомогательные окна, которые могли быть открыты (окна различных мастеров, вроде nntool, не закрываются). **Clear all** удаляет все переменные из области workspace, **clc** – очищает область окна Command window, где будет набираться код. **Format compact** устанавливает формат отображения для чисел, так для вещественных числе будет отображаться 4 точки после запятой.

Веса задаются обычным вектором **w = [4 -2]**, как видно, значения отделяются друг от друга пробелом, указывать тип не обязательно. Функция активации задаётся строкой **'tansig'**. Далее идёт вычисление взвешенной суммы: скалярное произведение входа на веса и прибавка смещения. Вектора **p** и **w** нельзя просто так взять и перемножить, т.к. по правилам линейной алгебры один из них должен быть транспонирован, что и делается с вектором **w** с помощью **w'**. Можно посмотреть значение переменной, оно должно быть -1. Далее вычисляем функцию активации при входе равном -1. Делаем это через **feval()**. Как понятно из названия, функция вычисляет значение некоторой функции (первый параметр) от вектора входов (второй параметр). Причём, как и многие другие фунции Matlab, она перегружена, т.е. её вызов может происходить при разных способах записи второго параметра. В первом случае вектор редуцируется к скалярной величине, выход равен -0.7616.

Далее присваиваем переменным **p1** и **p2** значения от -10 до +10 с шагом 0.25. Делаем это через **meshgrid()**, которая создаёт прямоугольную сетку в специальном формате пригодном для построения графиков. Но для построения самого графика нам нужно знать значения не только **p1** и **p2**, но и выхода нейрона. Теперь вычисляем эти значения для вектора **z** с помощью **feval()**, видно, что теперь второй параметр напрямую записывается в виде скалярного произведения двух векторов плюс смещение. Двоеточие означает перевод значения в формат double (используется для научных и инженерных вычислений. Никогда не используйте float для этих задач в таких языках как C/C++, чтобы избежать переполнения переменной).

Функция **reshape()** пересчитывает числовые значения в воксели для отображения на графике. **Plot3()** строит трёхмерный график, рисунок 5.1. **Grid on** включает сетку на этом графике, рисунок 5.2, 5.3. **Xlabel()**,

ylabel(), **zlabel()** – подписывают оси у графика.

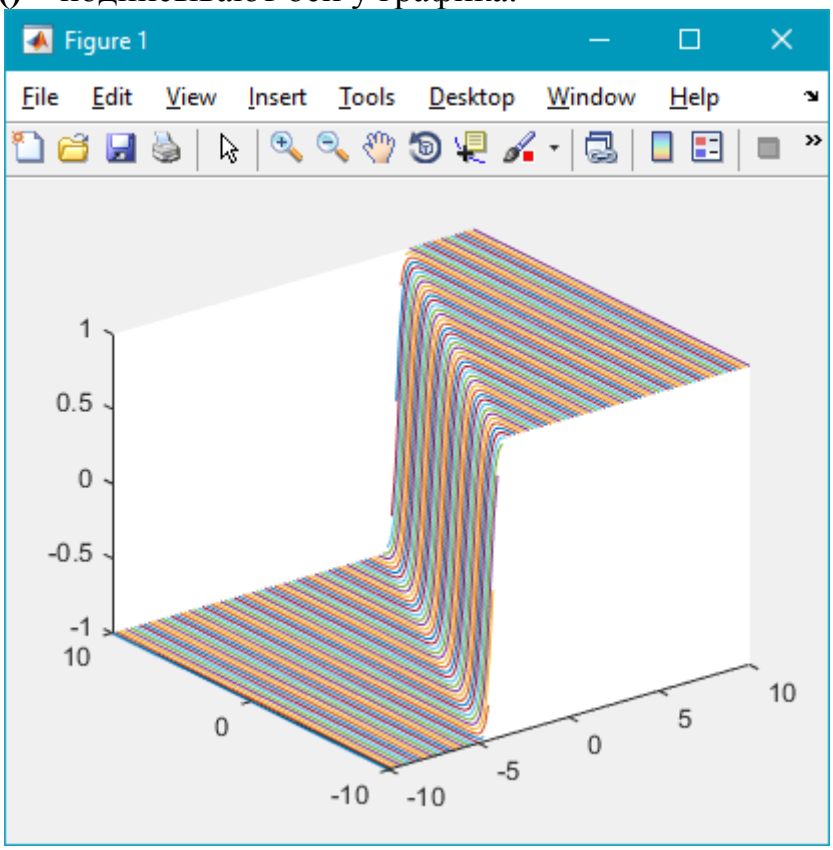


Рисунок 5.1 – Получившийся 3D-гарфик без сетки и подписей осей

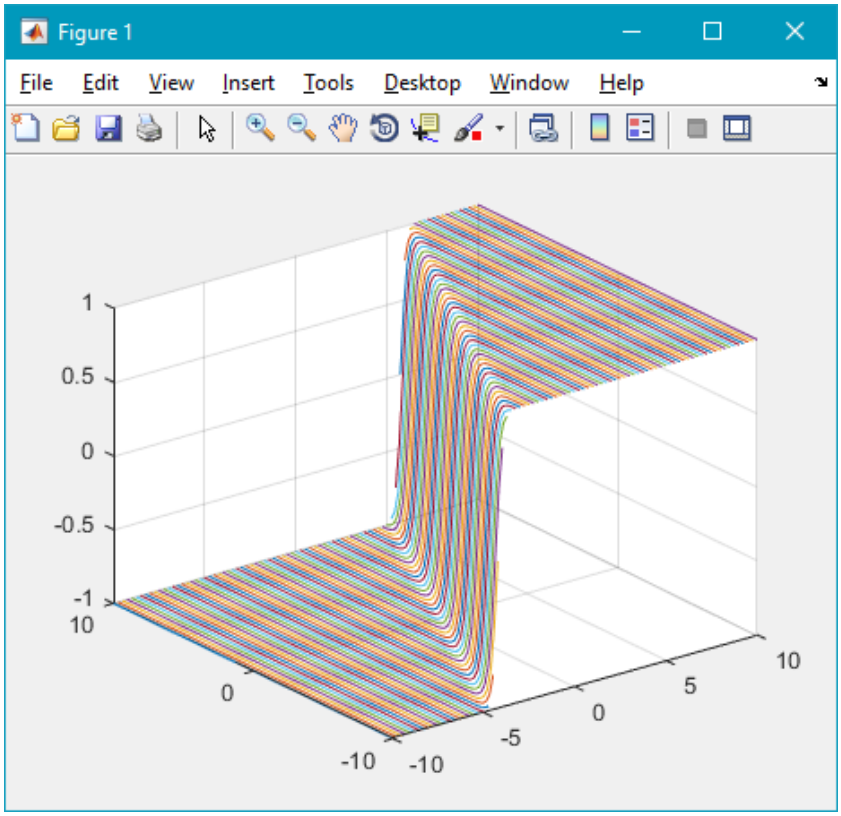


Рисунок 5.2 – Добавили сетку на фон

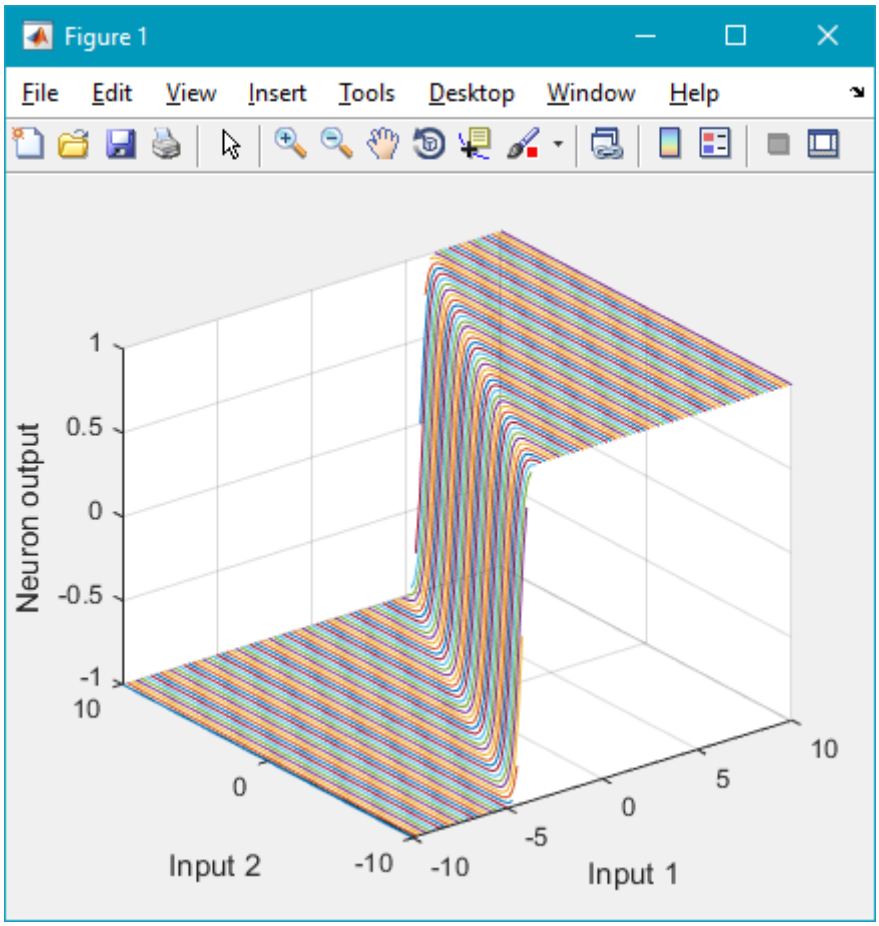


Рисунок 5.3 – Добавили подписи для осей *x*, *y*, *z*

Теперь создадим и обучим двухслойную сеть на одном примере.

Пример 2. Создание и обучение двухслойной сети.

```
close all, clear all, clc, format compact
% Входной транспонированный вектор
inputs = [1:6]'
% Выходной вектор, который должна промоделировать сеть
outputs = [1 2]';
% Задаём общую структуру сети
net = network (1, 2, [1; 0], [1; 0], [0 0; 1 0], [0 1]);
% Показываем её
view(net);
% Задаём количество промежуточных слоёв
net.layers{1}.size = 5;
% Задаём функции активации на них
net.layers{1}.transferFcn = 'logsig';
% Ещё раз отображаем сеть
view(net);
% Устанавливаем размерность входа и выхода
net = configure(net, inputs, outputs);
% Смотрим окончательный вариант сети
view(net);
% Получаем изначальный выход сети без обучения
initial_output = net(inputs);
% Устанавливаем метод обучения
```

```
net.trainFcn = 'trainlm';
% Устанавливаем функцию ошибки
net.performFcn = 'mse';
% Обучаем сеть на одном примере
net = train(net, inputs, outputs);
% Опять подаём вход, но уже на обученную сеть,
% сеть должна промоделировать выход [1 2]
final_output = net(inputs);
```

Входной вектор будет равен массиву из шести элементов от 1 до 6, т.к. диапазон записывается через «**:**». Функция **network()** задаёт общую структуру сети. Рассмотрим более подробно, что означает каждый параметр.

Первый параметр означает количество входов (не в смысле размерность входного вектора, а в смысле, – сколько векторов будет подано на вход сети. Допустим, в сверточных сетях обычное дело, что на вход подаётся не один, а два или три массива, каждый из которых как-то связан с дальнейшим слоем). У нас этот параметр равен 1. Второй параметр означает количество слоёв в сети. В данном случае имеем двухслойную сеть. Третий параметр – булев вектор, который означает какие нейроны будут иметь смещение, а какие – нет. Т.к. вектор [1; 0], то очевидно, что все нейроны первого слоя будут иметь смещение, а нейроны второго слоя – нет. Четвёртый параметр – это также булев вектор, который означает, – с какими слоями связан вход. В данном случае используется классическая схема, где вход связна только с последующим слоем, а, следовательно, вектор [1; 0]. Пятый параметр – это матрица связей между слоями, которая означает, какой слой с каким связан. Она записывается в виде вектора. Имеем [0 0; 1 0], если записать в виде матрицы, то получится $\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$. Тогда понятно, что первый слой связан со вторым слоем, т.к. единица располагается в позиции (1, 2), где 1 – это номер столбца, 2 – номер строки. Шестой параметр [0 1] – булев вектор, который показывает, с какими слоями связан выход. При таких значениях выход связан с предпоследним слоем.

При такой конфигурации получим структуру сети как на рисунке 5.4.

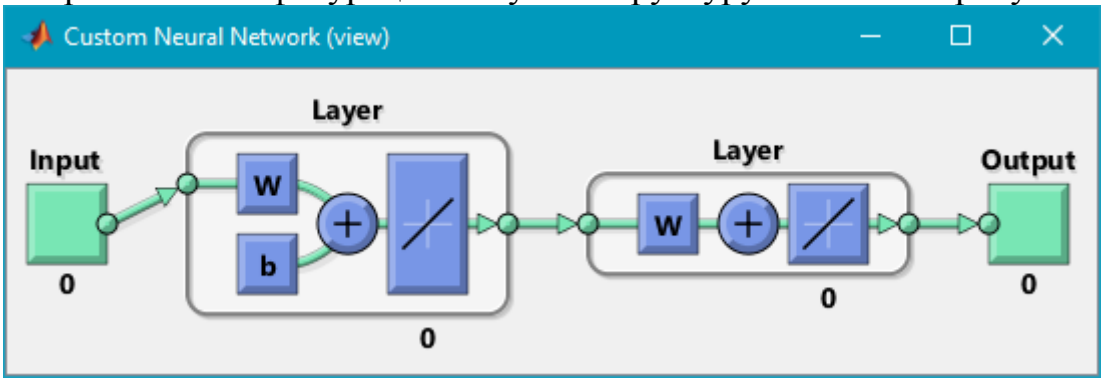


Рисунок 5.4 – Изначальная структура сети

Изменим третий параметр с [1;0] на [1;1], получим структуру как на рисунке 5.5.

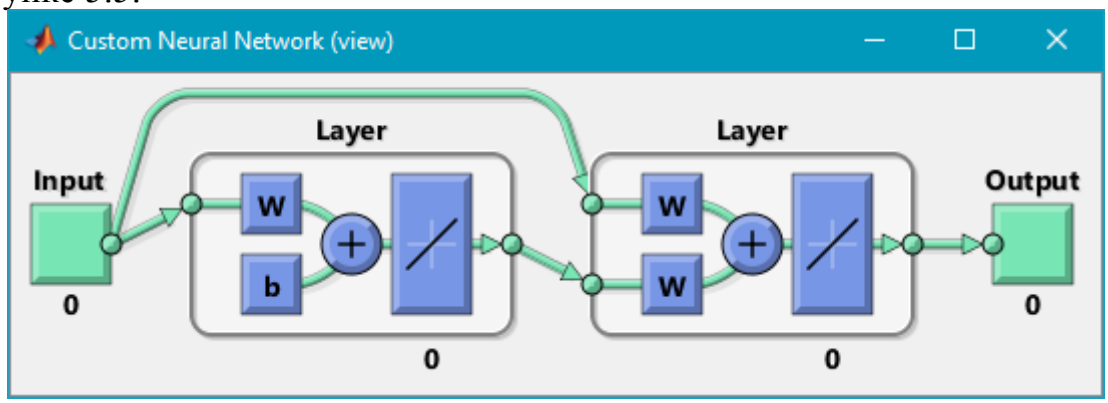


Рисунок 5.5 – Видно, что у второго слоя тоже появилось смещение

Изменим теперь и четвёртый параметр на [1; 1], получим структуру сети как на рисунке 5.6.

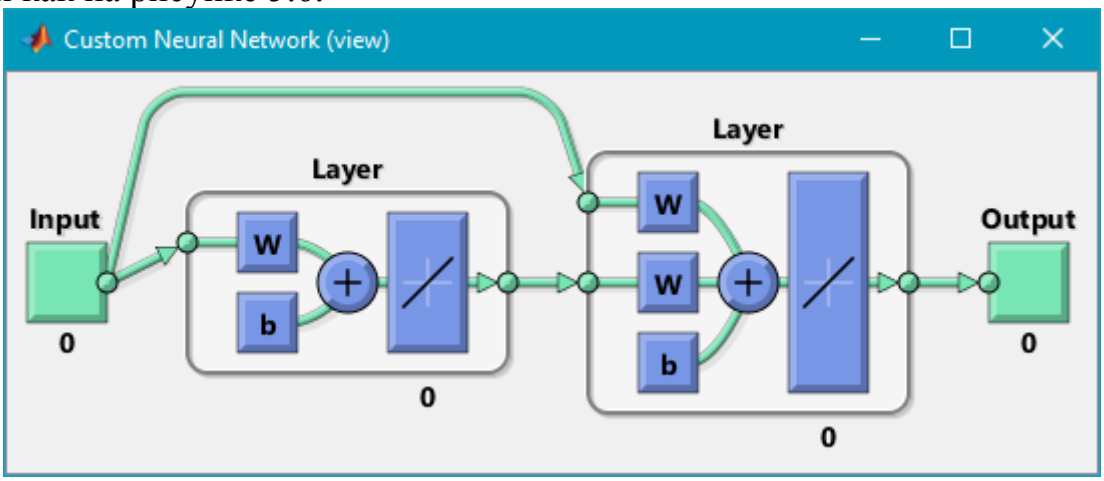


Рисунок 5.6 – Добавилась связь входа со вторым слоем

Изменим остальные два вектора на единичные, получим структуру как на рисунке 5.7.

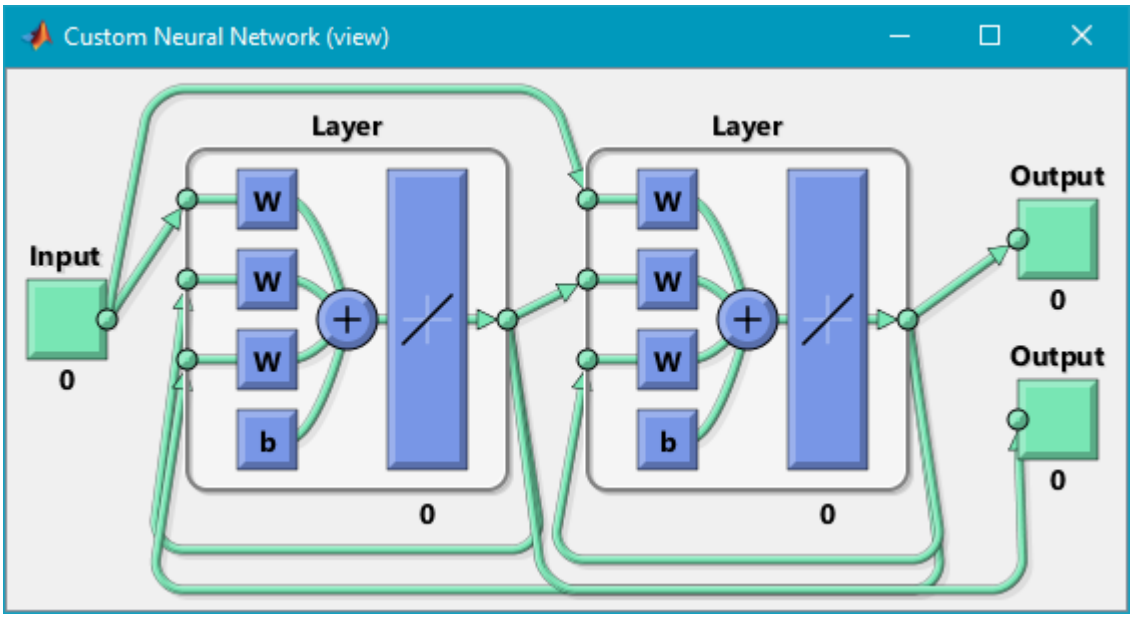


Рисунок 5.7 – Появился второй выход, связанный с первым слоем, а также каждый слой связан с самим собой и второй слой связан с первым (подаёт свои сигналы на вход первого слоя)

Net.layers{1}.size = 5 – присваиваем количество нейронов первому слою сети (индекс записывается в фигурных скобках). Видно, что обращение к данным параметрам осуществляется классическим образом для объектов: через точку. Т.е. сеть в Matlab – это объект (по терминологии объектно-ориентированного программирования (ООП)).

Небольшое примечание. Концепция ООП напрашивается для описания ИНС, т.к. вполне логично, что объект «сеть» включает в себя объекты – «слой», а те – «нейрон». Однако, нужно помнить, что при таком подходе оперативная память выделяется и группируется под конкретные объекты и в случае попыток быстрой реализации ИНС, возможно, с использованием ассемблера, не получится быстро реализовывать матричные операции. Короче, если требуется быстрая реализация сети, то от ООП-описания придётся отказаться.

Net.layers{1}.transferFcn = 'logsig' – присваивает слоям первого слоя функцию активации сигмойд (без отрицательных значений), рисунок 3.8.

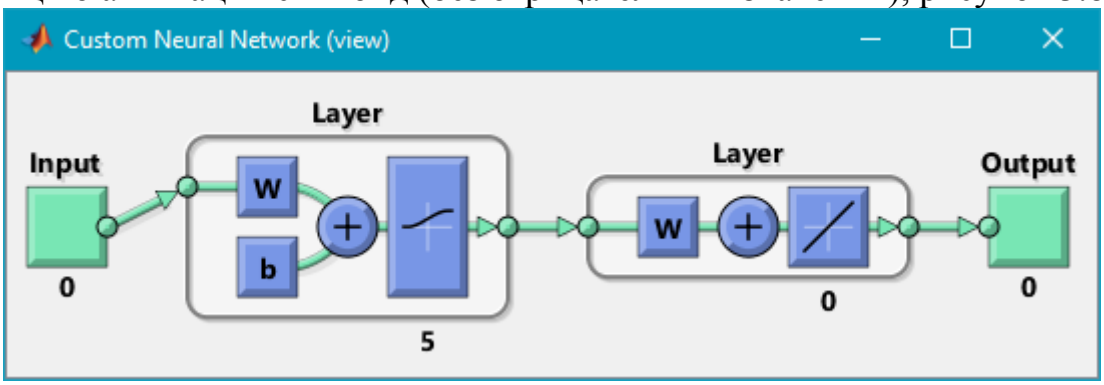


Рисунок 5.8 – Меняем функцию активации у нейронов первого слоя

Для установки входного и выходного слоя можно сконфигурировать сеть по отношению к вектору входа и выхода: **net = configure(net, inputs,**

outputs), рисунок 5.9.

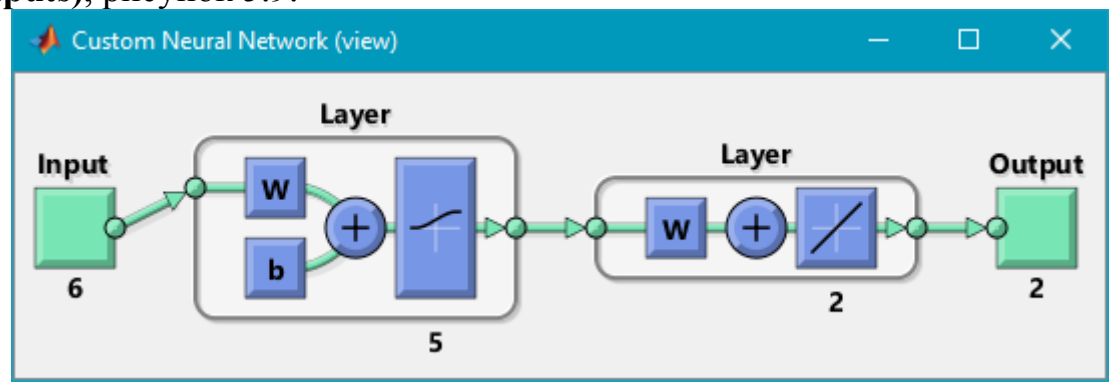


Рисунок 5.9 – Окончательно получаем общую структуру сети, где размер входа – 6, а выхода – 2

Вектор **initial_output** получит значения $[0\ \ 0]^T$, т.к. веса не инициализированы, т.е. сеть ничего делать не умеет. Для начала обучения нужно как минимум задать метод обучения и функцию ошибки, что и делается с помощью **net.trainFcn = 'trainlm'** и **net.performFcn = 'mse'**. **Trainlm** – это обучение по методу Левенберга-Марквардта, а **MSE** (mean square error) – это среднеквадратическая ошибка. **Train()** – запускает обучение сети и возвращает объект – обученную сеть. Процесс обучения показан на рисунке 5.10.

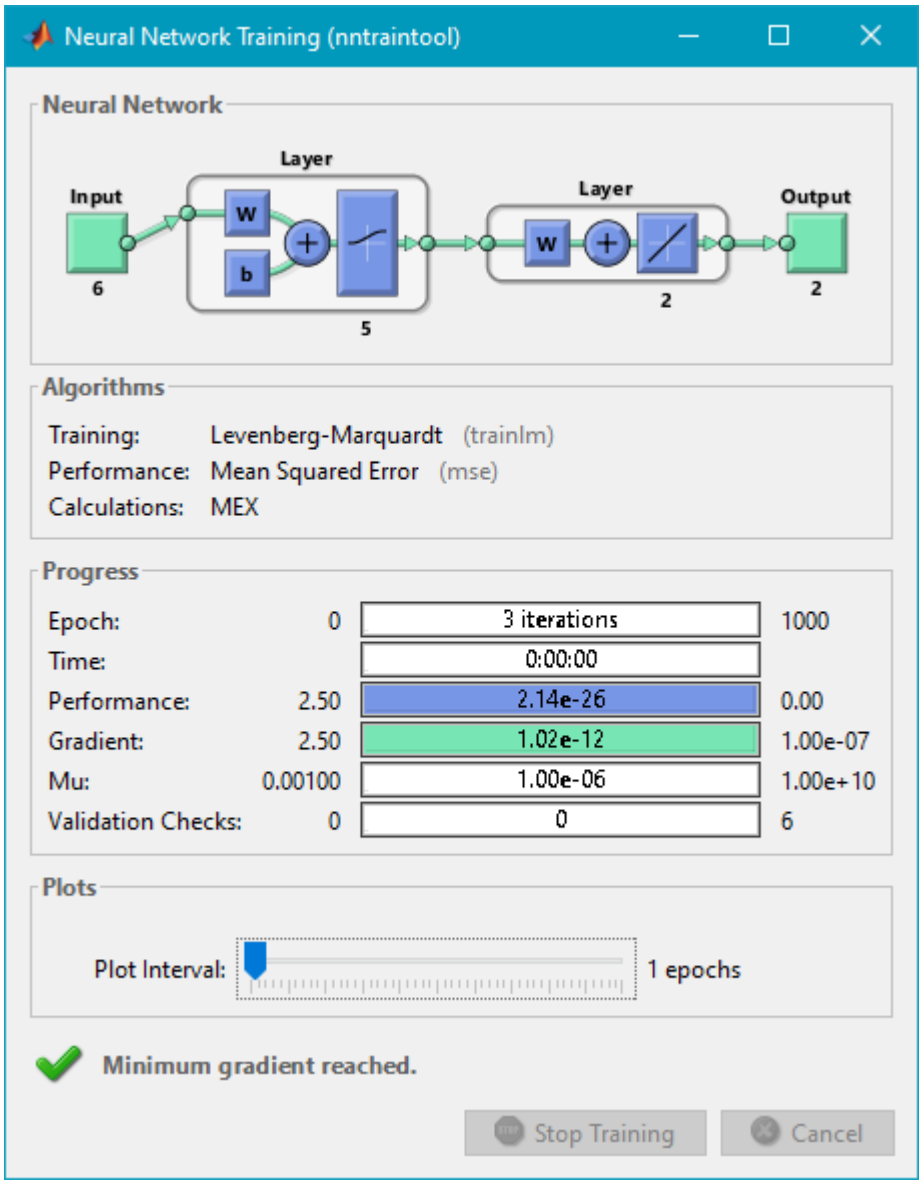


Рисунок 5.10 – Обучение сети

Позже формируем вектор **final_output**, который содержит ответ сети на вход, т.е. 1.0000 и 2.0000 (помним про формат **compact**).

Теперь решим задачу о разделении классов с помощью персептрона, подобную той, которая была решена в лабораторной работе 3.

Пример 3. Разделение классов с помощью персептрона.

```
close all, clear all, clc, format compact
% Задаём количество паттернов каждого класса
N = 20;
% Задаём смещение
offset = 5;
% Создаём входные значения каждого класса
x = [randn(2, N) randn(2, N)+offset];
% Создаём выходные значения для этих классов
y = [zeros(1, N) ones(1, N)];
% Открываем окно для рисования
figure(1);
```

```
% Наносим точки (паттерны)
plotpv(x, y);
% Создаём объект-сеть типа Персептрон
net = perceptron;
% Обучаем персептрон
net = train(net, x, y);
% Рисуем общую структуру сети
view(net);
% Возвращаемся в окно для рисования и рисуем прямую линию,
% которая разделит два класса
figure(1);
plotpc(net.IW{1}, net.b{1});
```

rand(2, N) – создаёт массив размером 2xN, числа которого являются случайные величины, распределённые по нормальному закону с математическим ожиданием 0 и среднеквадратическим отклонением 1. Таким образом, будут созданы две матрицы 2xN, содержащие координаты точек-паттернов в двумерном пространстве, причём координаты второй матрицы будут смещены на 5 позиций, т.е. заранее гарантируется, что классы будут линейно разделимые. Обратим внимание, что эти две матрицы на самом деле записаны в одну матрицу размером 2x2*N. **Zeros(1, N)** и **ones(1, N)** создаёт массив нулей и единиц размером N, это метки для входных паттернов.

Figure(1) – вызывает окно, содержащее канву для рисования, рисунок 5.11.

Plotpv(x, y) – наносит точки на канву с координатами x и метками y, рисунок 5.12. Причём нули отображаются кружками, а единицы плюсами.

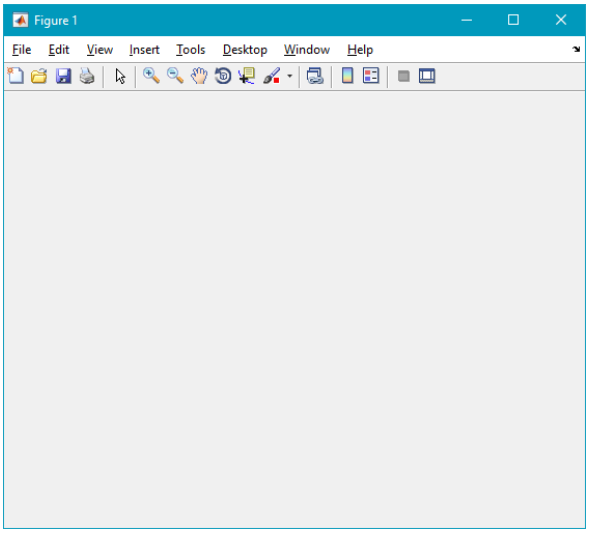


Рисунок 5.11 – Окно для рисования

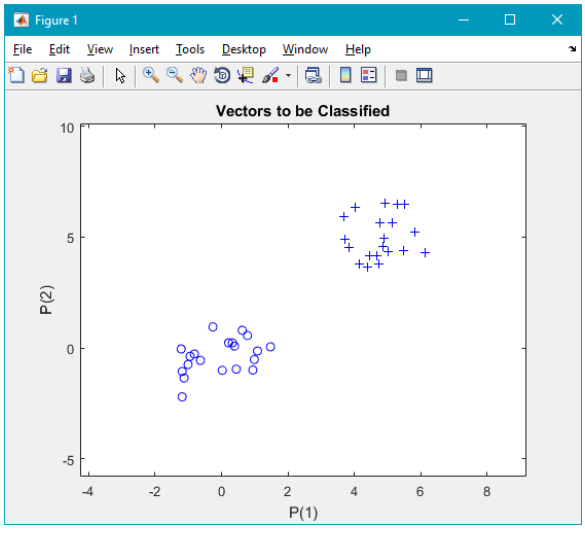


Рисунок 5.12 – Два созданных нами класса, видно, что они линейно разделимы

Net = perceptron – создаём сеть типа Персептрон, если после записи команды не ставить точку с запятой, то среда развернёт все установленные по умолчанию параметры, рисунок 5.13.

```
Command Window
>> net=perceptron
net =
    Neural Network
              name: 'Perceptron'
          userdata: (your custom info)
    dimensions:
         numInputs: 1
         numLayers: 1
        numOutputs: 1
    numInputDelays: 0
    numLayerDelays: 0
 numFeedbackDelays: 0
 numWeightElements: 0
        sampleTime: 1
    connections:
       biasConnect: true
      inputConnect: true
      layerConnect: false
     outputConnect: true
    subobjects:
              input: Equivalent to inputs{1}
             output: Equivalent to outputs{1}
             inputs: {1x1 cell array of 1 input}
             layers: {1x1 cell array of 1 layer}
            outputs: {1x1 cell array of 1 output}
             biases: {1x1 cell array of 1 bias}
       inputWeights: {1x1 cell array of 1 weight}
       layerWeights: {1x1 cell array of 0 weights}
    functions:
           adaptFcn: 'adaptwb'
         adaptParam: (none)
           derivFcn: 'defaultderiv'
          divideFcn: 'dividetrain'
        divideParam: (none)
         divideMode: 'sample'
            initFcn: 'initlay'
```

Рисунок 5.13 – Методы и свойства объекта perceptron

Далее происходит обучение сети. Общая структура сети представлена на рисунке 5.14.

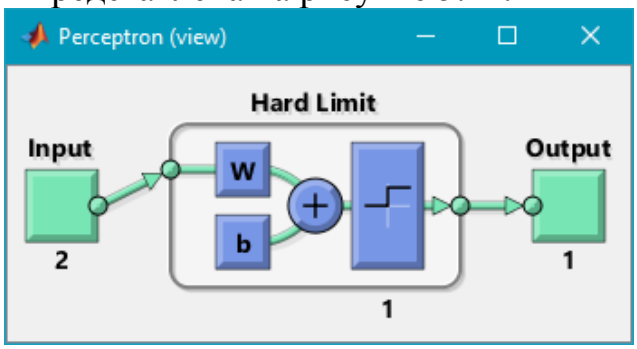


Рисунок 5.14 – Общая структура сети

Результаты обучения представлены на рисунке 5.15. Видно, что потребовалось 29 итераций.

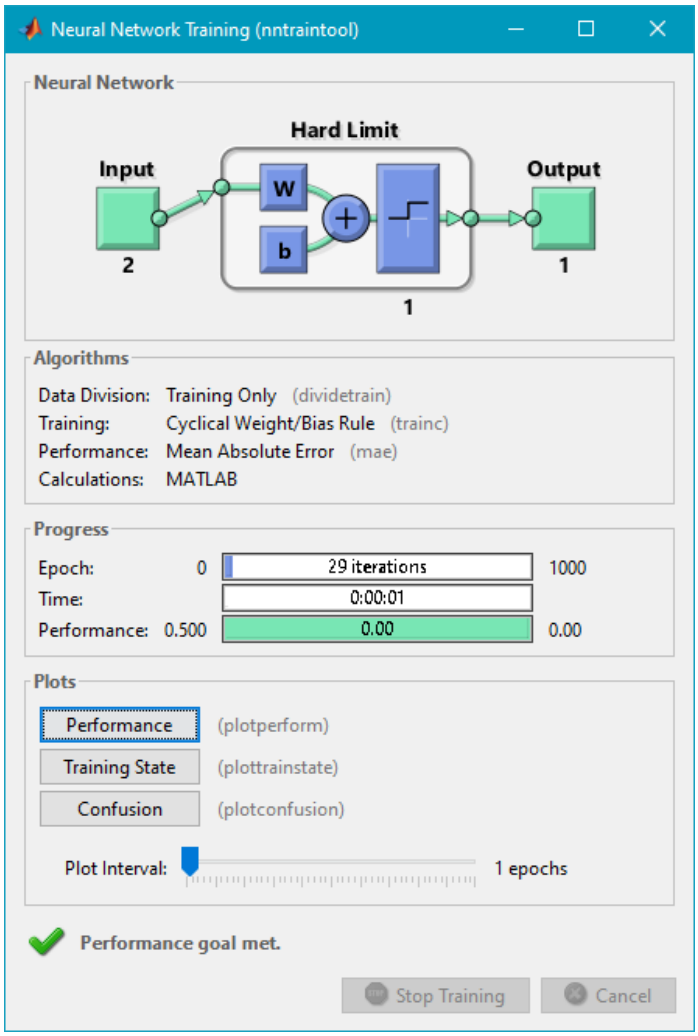


Рисунок 5.15 – Результаты обучения персептрона

На рисунке 5.16 представлен график настройки персептрона. График носит характерную зубчатую форму. После каждой эпохи высчитывается MSE между предполагаемыми ответами и реальными. Напомним, что при адаптации не используется спуск по поверхности ошибки, поэтому ждать постепенного снижения ошибки обучения не следует. Обучение идёт до тех пор, пока не будет найдена первая прямая, отделяющая один класс от другого.

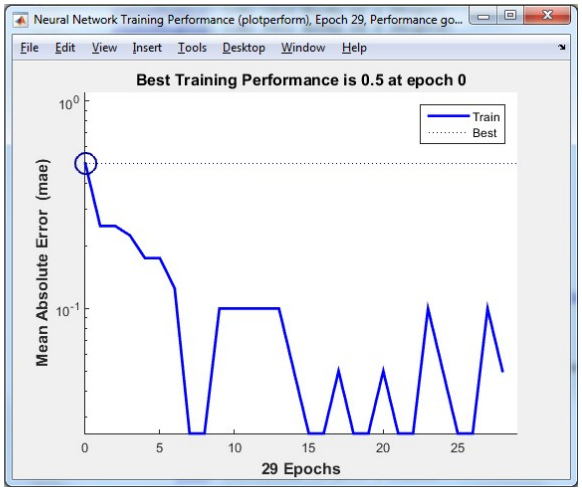


Рисунок 5.16 – График настройки весовых коэффициентов персептрона

Plotpc(net.IW{1}, net.b{1}) – строит прямую линию с соответствующими коэффициентами. **IW** – обозначает слой коэффициентов между входом и первым слоем, **b{1}** – смещение первого слоя. Итоговая прямая показана на рисунке 5.17.

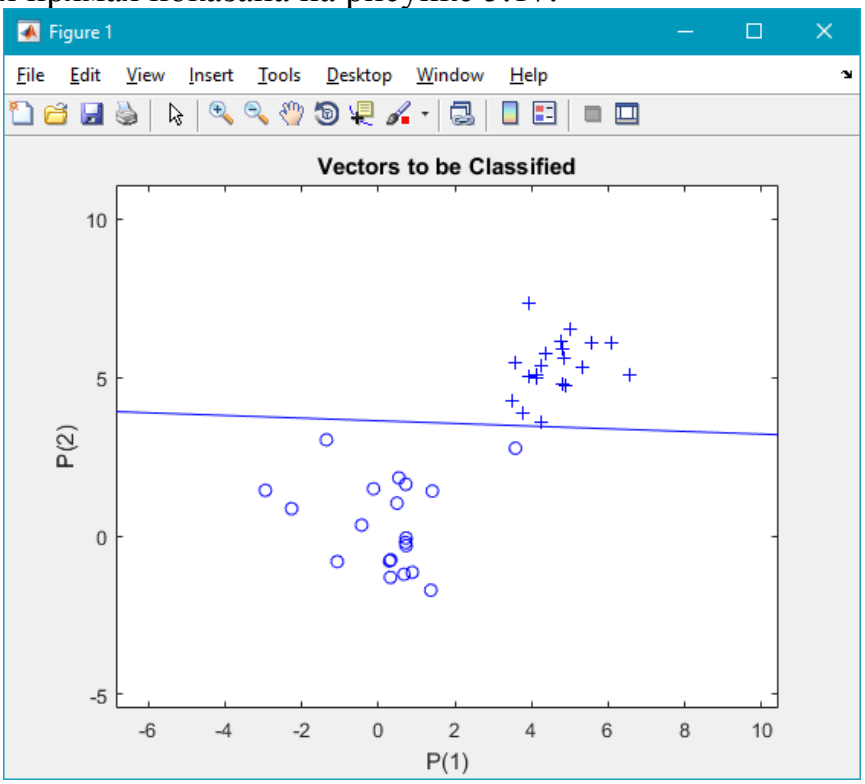


Рисунок 5.17 – Полученное решение по разделению двух классов

Теперь рассмотрим решение задачи разделения на 4 класса с помощью персептрона и обычной многослойной сети прямого распространения. Расположение классов оставим таким же, как и в задаче XOR (по углам «квадрата»), но теперь у нас не два класса, а 4, а также каждый класс представлен 30 паттернами.

Пример 4. Решение задачи разделения на 4 класса с помощью персептрона и обычной многослойной сети прямого распространения.

```
close all, clear all, clc, format compact
    % Каждый класс имеет 30 паттернов
    K = 30;
    % Смещение для классов 0.6
    q = .6;
    % Задаём 4 класса в виде векторов A, B, C, D (2 строки и 30 столбцов)
    A = [rand(1, K)-q; rand(1, K)+q];
    B = [rand(1, K)+q; rand(1, K)+q];
    C = [rand(1, K)+q; rand(1, K)-q];
    D = [rand(1, K)-q; rand(1, K)-q];
    % Рисуем на канве точки A определённого типа (третий параметр)
    plot(A(1, :), A(2, :), 'bs');
```

```
% Фиксируем канву
hold on;
grid on;
% На той же канве рисуем остальные классы
plot(B(1, :), B(2, :), 'r+');
plot(C(1, :), C(2, :), 'go');
plot(D(1, :), D(2, :), 'm*');
text(.5-q, .5+2*q, 'Class A');
text(.5+q, .5+2*q, 'Class B');
text(.5+q, .5-2*q, 'Class C');
text(.5-q, .5-2*q, 'Class D');
% Задаём кодировку классов (метки)
a = [0 1]';
b = [1 1]';
c = [1 0]';
d = [0 0]';
% Получаем общий вектор входных значений
P = [A B C D];
% Получаем общий вектор меток
T = [repmat(a, 1, length(A)) repmat(b, 1, length(B)) repmat(c, 1, length(B)) repmat(d, 1, length(D))];
% Сеть типа «персептрон»
net = perceptron;
% Пусть среднеквадратическая ошибка не равна 0
E = 1;
% Установка параметра о том, что будет минимум один прогон
net.adaptParam.passes = 1;
% Построили изначально ответ персептрона, до обучения
linehandle = plotpc(net.IW{1}, net.b{1});
% Счётчик по циклам
n = 0;
% Цикл будет идти пока среднеквадратическая ошибка не
% станет равной нулю или пока не достигнем значения итераций в 1000
while (sse(E) & n<1000)
    n = n + 1;
% Обучить персептрон
    [net, Y, E] = adapt(net, P, T);
% Нарисовать линии
linehandle = plotpc(net.IW{1}, net.b{1}, linehandle);
drawnow;
end
view(net);
% Проверка работы персептрона на входном векторе
p = [0.7; 1.2];
% Выдаст класс (1; 1)
y = net(p);
```

rand(1, K) – генерирует числа из равномерного распределения в виде вектора (первый параметр) размером K, в итоге **A = [rand(1, K)-q; rand(1, K)+q]** – генерируется матрица 2xK, что соответствует координатам точек.

Plot(A(1, :), A(2, :), 'bs') – рисует на канве точки из матрицы A, переводя их в вещественный формат. Параметр **'bs'** – это цвет и форма

точек: b – синий и s – квадрат. **'r+'** – красный (r) крест (+), **'go'** – зелёный (g) кружок (o), **'m*'** – пурпурная (m) звездочка (*).

Text(.5-q, .5+2*q, 'Class A') – наносит текст на канву с координатами, записанными как два первых параметра.

Общий вид отображения классов представлен на рисунке 5.18.

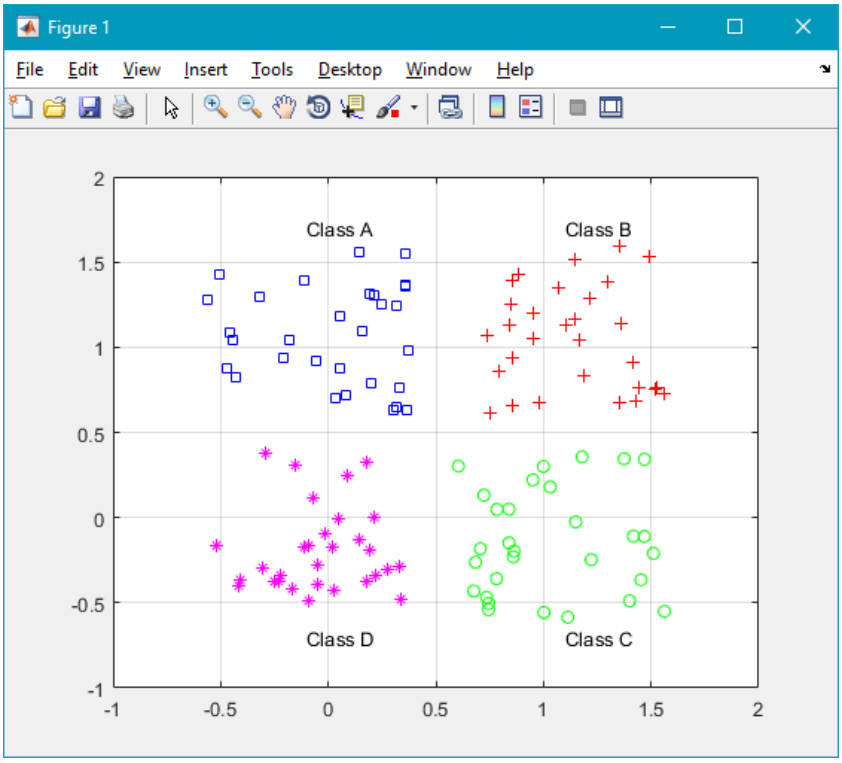


Рисунок 5.18 – Классы, которые предстоит разделить

Для получения вектора T необходимо раскопировать транспонированные вектора **a**, **b**, **c**, **d**. Для этой цели используется **repmat()**: **repmat (что копируем, копий по строкам, копий по столбцам)**. В результате вектор **T** примет вид как на рисунке 3.19: матрица из двух строк и 30 * 4 = 120 колонок.

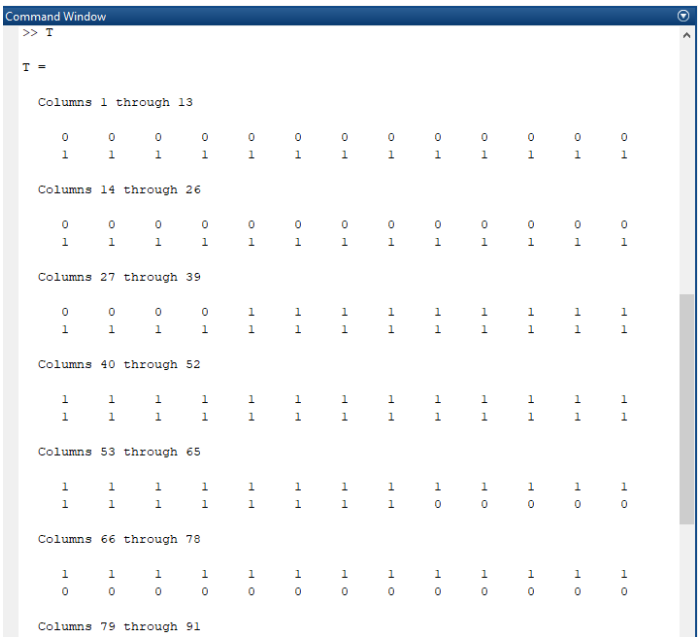


Рисунок 5.19 – Общий вид вектора меток T

Цикл **while()** – это цикл с предусловие, поэтому он может не выполниться ни разу, а следовательно, перед ним необходимо построить возможное решение (т.к. при создании сети **net = perceptron**) веса получают случайные значения.

[net, Y, E] = adapt(net, P, T) – процесс адаптации весов. **Net** – получит обученную сеть, **Y** – вектор ответов сети для входа **P**, **E** – ошибки сети для меток **T** и входов **P**. **Sse(E)** – сумма квадратов этих ошибок.

На рисунке 3.20 показана структура сети, видно, что единственный слой сети содержит два нейрона, что соответствует двум прямым линиям для разделения 4-х классов.

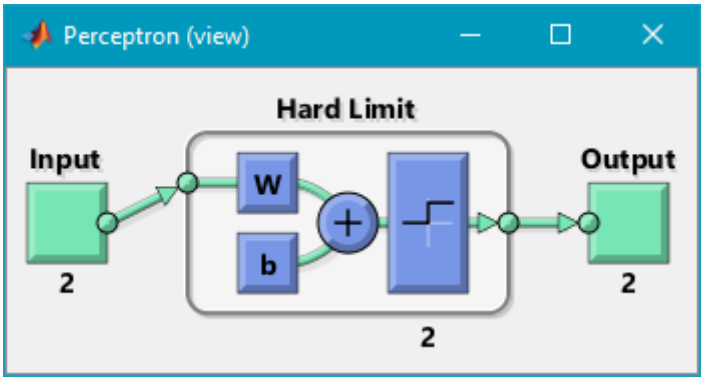


Рисунок 5.20 – Общая структура персептрона

Возможный ответ сети приведён на рисунке 5.21.

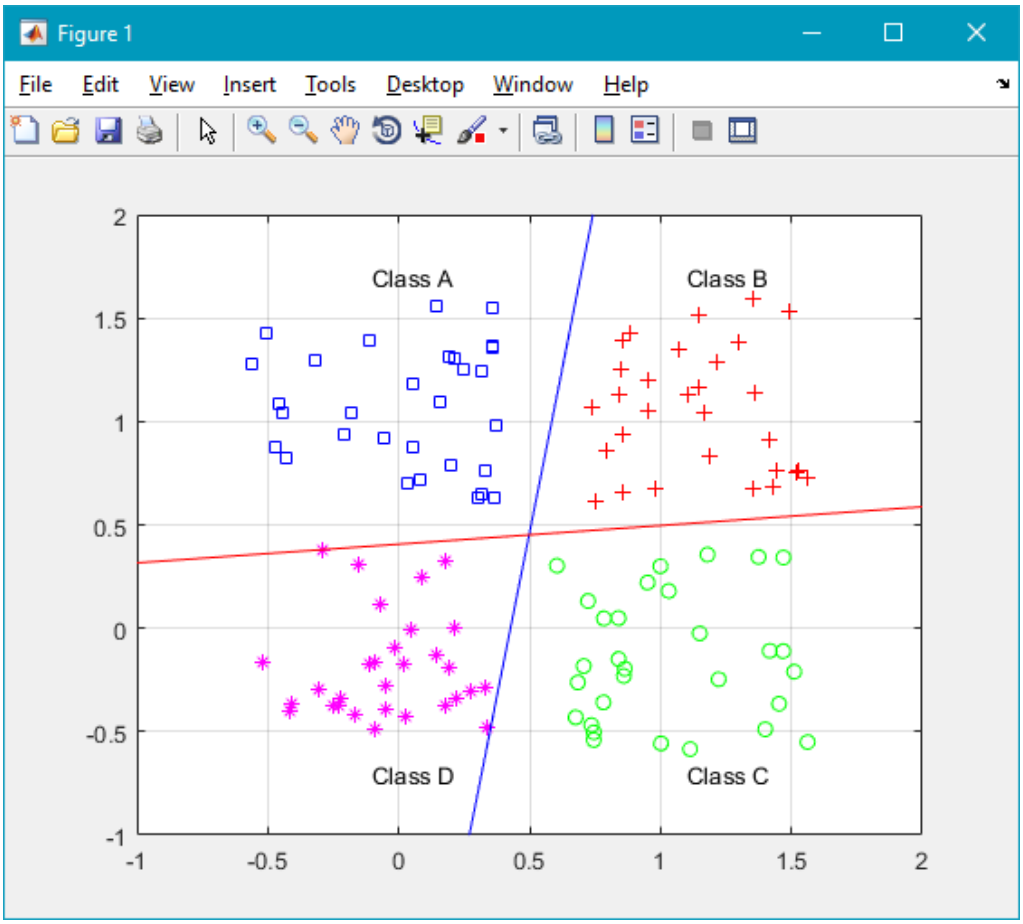


Рисунок 5.21 – Решения задачи разбития на 4 класса

Рассмотрим решение XOR-проблемы с помощью многослойной сети прямого распространения, но расширим представление каждого класса до 100 паттернов.

Пример 5. Решение примера 4 с помощью многослойной сети прямого распространения**.**

```
close all, clear all, clc, format compact
K = 100;
q = .6;
% Данные для обучения
%Обратите внимание на то, что randn() отделяются с помощью
%";". Если опустить этот символ, то A будет не 2x100, а 1x200.
A = [rand(1, K)-q; rand(1, K)+q];
B = [rand(1, K)+q; rand(1, K)+q];
C = [rand(1, K)+q; rand(1, K)-q];
D = [rand(1, K)-q; rand(1, K)-q];
figure(1);
% Наносим точки на канву
plot(A(1, :), A(2, :), 'k+');
hold on;
grid on;
plot(B(1, :), B(2, :), 'bd');
plot(C(1, :), C(2, :), 'k+');
plot(D(1, :), D(2, :), 'bd');
% Наносим названия классов
```

```
text(.5-q, .5+2*q, 'Class A');
text(.5+q, .5+2*q, 'Class B');
text(.5+q, .5-2*q, 'Class A');
text(.5-q, .5-2*q, 'Class B');
% Определяем метки для классов
a = -1, c = -1, b = 1, d = 1;
% Формируем входной вектор
P = [A B C D];
% Формируем вектор ответов
T = [repmat(a, 1, length(A)) repmat(b, 1, length(B))   repmat(c, 1, length(C))
repmat(d, 1, length(D))];
% Определяем сеть прямого распространения
net = feedforwardnet([5 3]);
% Вся выборка под обучающие примеры
net.divideParam.trainRatio = 1;
% Нет валидационной выборки
net.divideParam.valRatio = 0;
% Нет тестовой выборки
net.divideParam.testRatio = 0;
% Обучаем сеть
[net, tr, Y, E] = train(net, P, T);
view(net);
figure(2);
% График ожидаемых ответов сети
% график получается зубчатым, т.к. метки для классов
%определены как -1 и +1
plot(T', 'linewidth', 2);
hold on;
% График реальных ответов сети
plot(Y', 'r--');
grid on;
% Рисуем легенду к графикам
legend('Targets', 'Network response', 'location', 'best');
% Определяем видимый диапазон по оси y
ylim([-1.25 1.25]);
% Создаём  набор 601 числа
span = -1:.005:2;
% P1 и P2 станут матрицами размером 601x601
[P1, P2] = meshgrid(span, span);
% Транспонируем вектор размером 601*601 = 361201
pp = [P1(:) P2(:)]';
% Получаем выходы сети
aa = net(pp);
% Возвращаемся к первой канве
figure(1);
% Рисуем трёхмерную сетку
mesh(P1, P2, reshape(aa, length(span), length(span))-5);
% Подбираем лучшее сочетание цветов
colormap cool;
```

На рисунке 5.22 показаны паттерны входных классов.

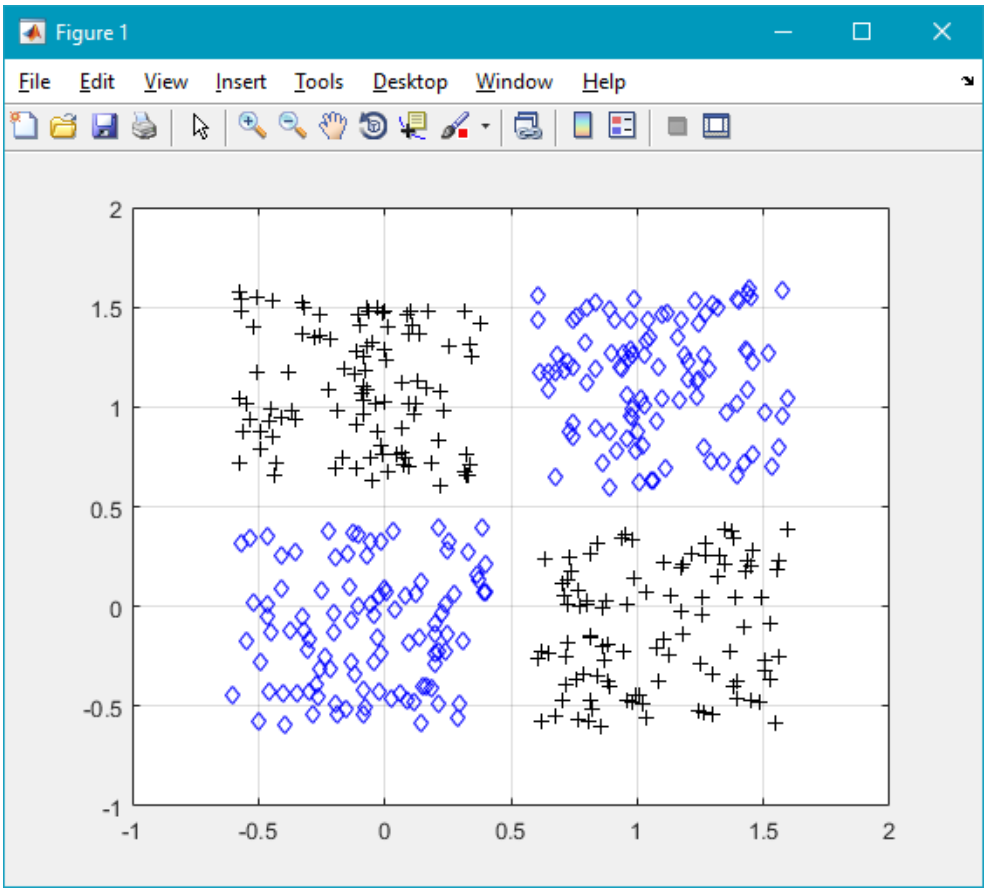


Рисунок 5.22 – Входные данные для двух классов, которые нужно разделить

Net = feedforwardnet([5 3]) – создаём сеть прямого распространения с двумя слоями, в первом будет 5 нейронов, во втором – 3. Структура сети показана на рисунке 3.23.

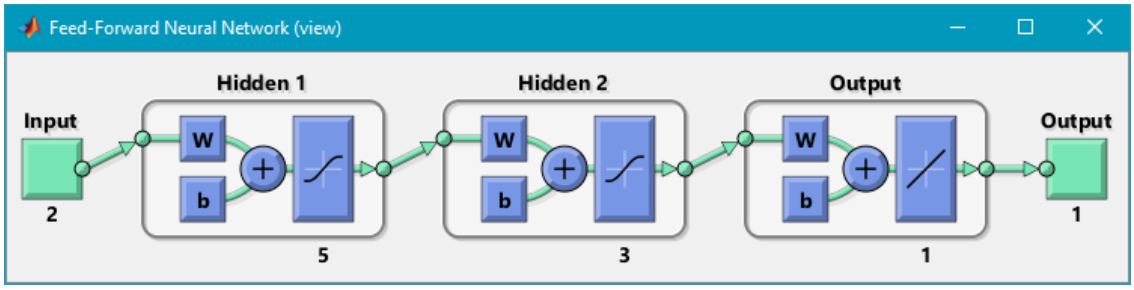


Рисунок 5.23 – Общая структура сети

Обратим внимание, что выходной нейрон имеет линейную функцию активации.

Все паттерны войдут в обучающую выборку, поэтому **net.divideParam.trainRatio = 1**, валидационную и тестовую выборку установим в 0.

Обучение сети осуществляется через **[net, tr, Y, E] = train(net, P, T)** – где **tr** – переменная, содержащая информацию об обучении (в среде Matlab её можно открыть и посмотреть на входящие в неё поля и их значения). Процесс обучения показан на рисунке 5.24, кривая обучения на

рисунке 5.25.

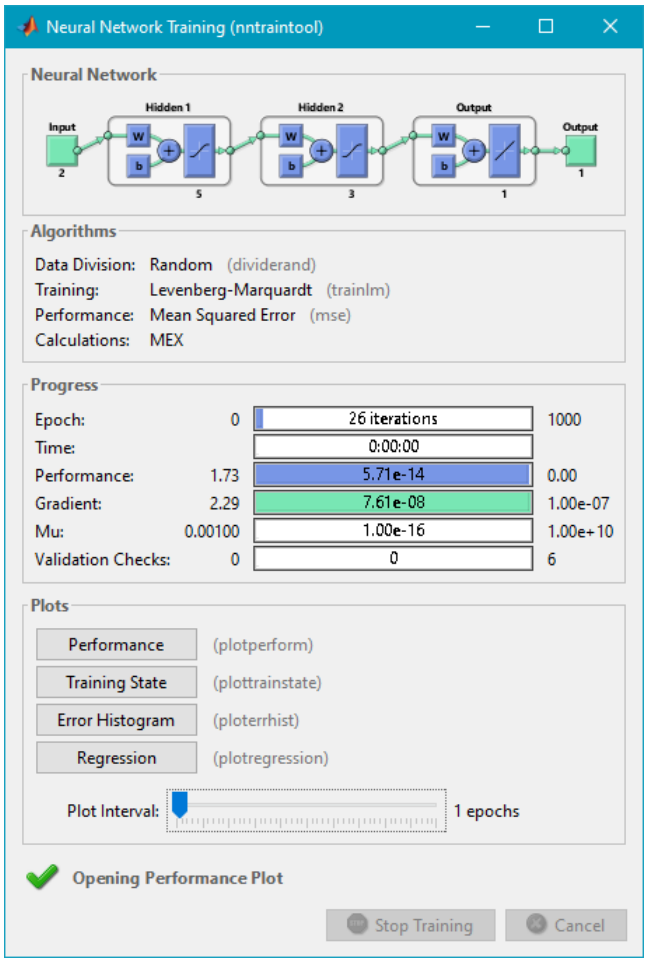


Рисунок 5.24 – Процесс обучения сети, потребовалось 26 итераций

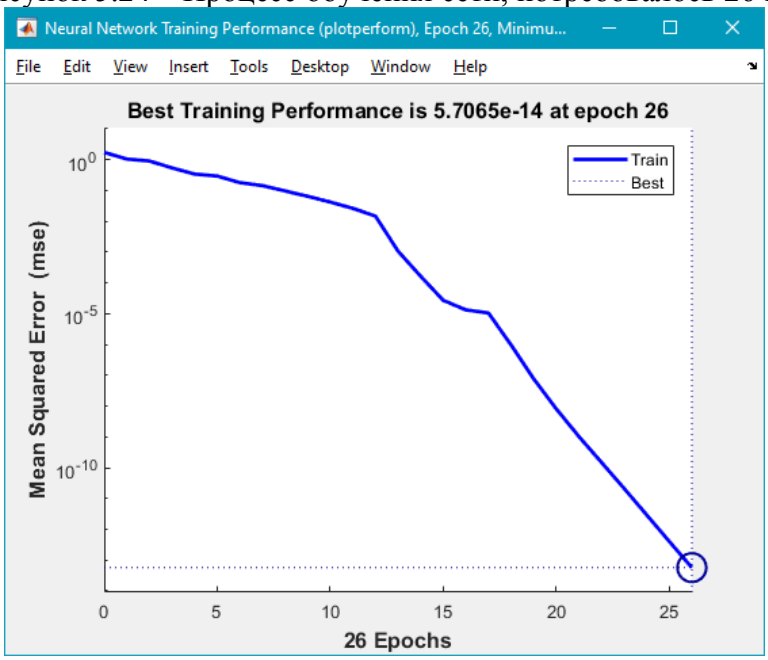


Рисунок 5.25 – Кривая обучения (MSE), лучший показатель меньше 10^{-10}

Далее отображаем на новой канве ожидаемый ответ сети, рисунок 5.26.

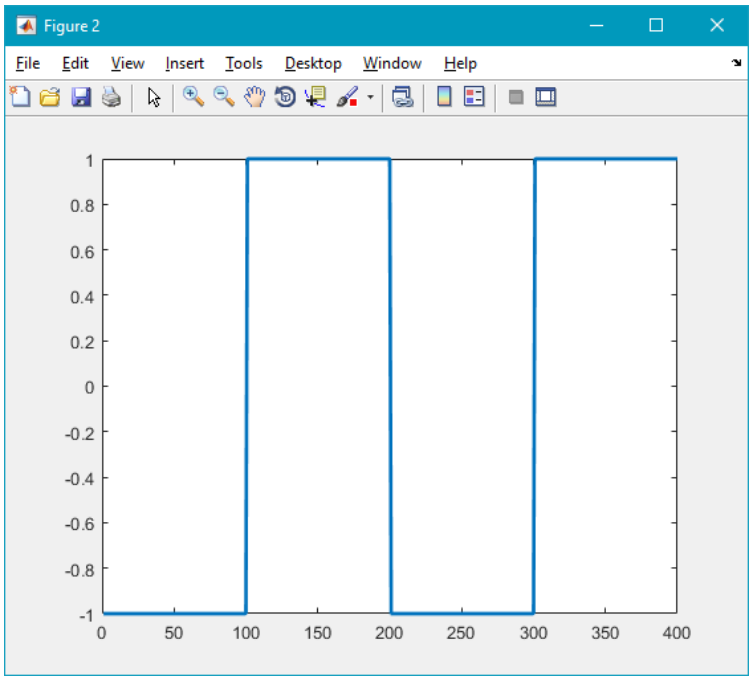


Рисунок 5.26 – Ожидаемый ответ сети (классы кодировались ±1)

На той же канве отображаем реальный ответ сети, рисунок 5.27. Видно, что графики совпадают.

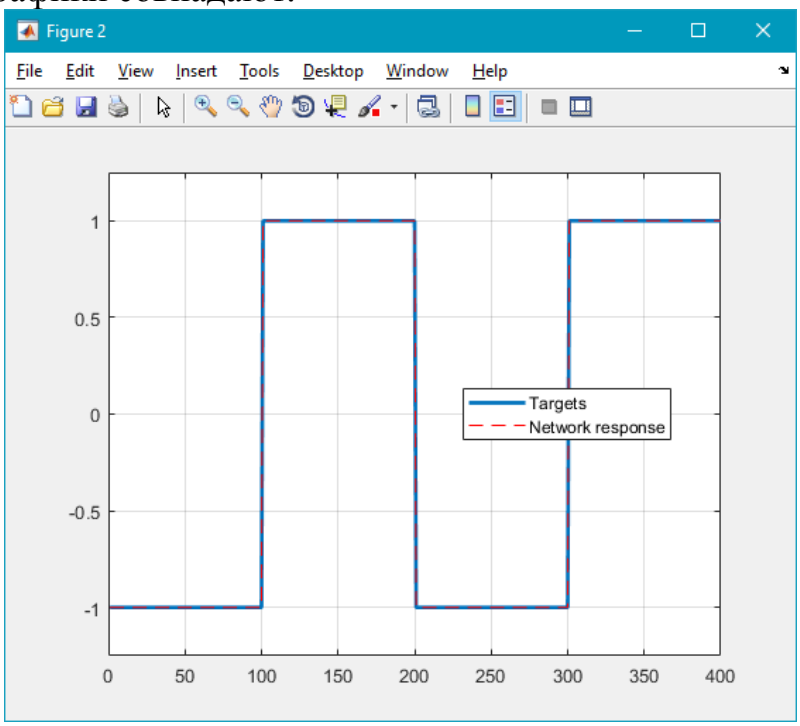


Рисунок 5.27 – Реальный ответ сети, графики совпадают

Ylim([-1.25 1.25]) – ограничивает ось OY данными значениями.

Чтобы отобразить получившееся решение уже нельзя просто построить прямые линии (как в случае с персептроном). Для этого нужно поверх первой канвы отобразить значения выходов сети на очень мелкой сетке (входные данные) и закрасить получившиеся ответы. Граница решения для разбиения двух классов окажется видна, рисунок 5.28.

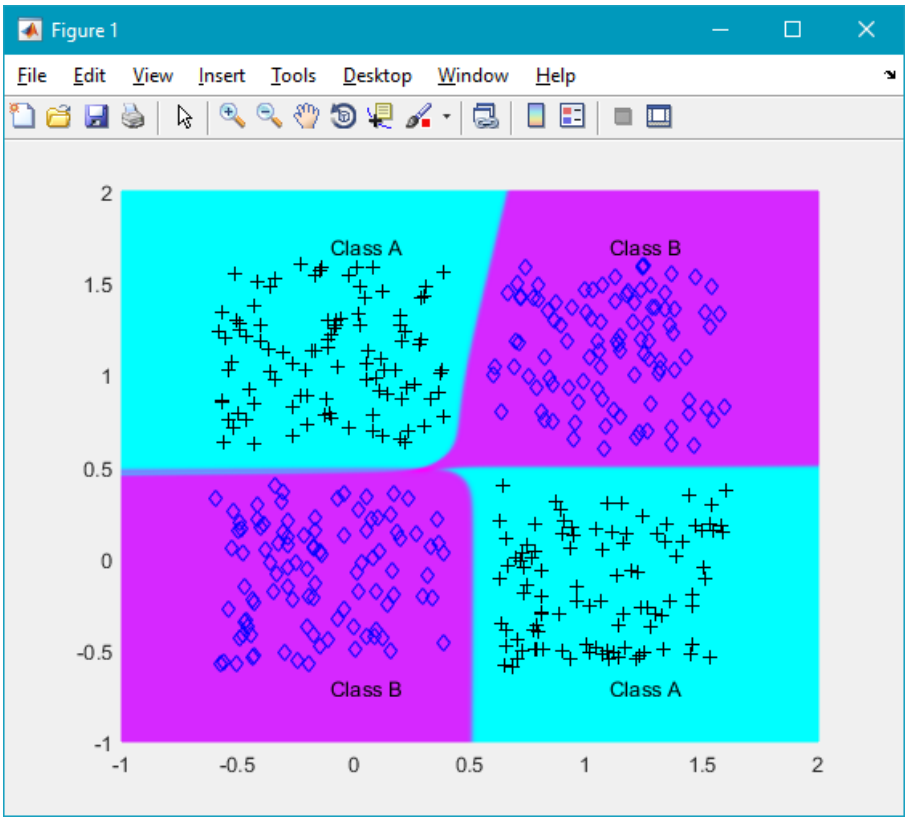


Рисунок 5.28 – Получившаяся граница решения для задачи XOR

Примечание. Сеть со структурой 2-5-3-1 явно избыточна для решения данной задачи.

Пример 6. Переопределение структуры сети.

Можно переопределить структуру сети, как было показано в начале данной лабораторной работы:

```
net.layers{1}.size = 5;
% Получаем двухслойную сеть, где скрытый слой имеет 5 нейронов
net = configure(net, P, T);
% Меняем функцию активацию на выходном нейроне на нелинейную
net.layers{2}.transferFcn = 'tansig';
net.divideParam.trainRatio = 1;
net.divideParam.valRatio = 0;
net.divideParam.testRatio = 0;
[net, tr, Y, E] = train(net, P, T);
```

Структура такой сети показана на рисунке 5.29.

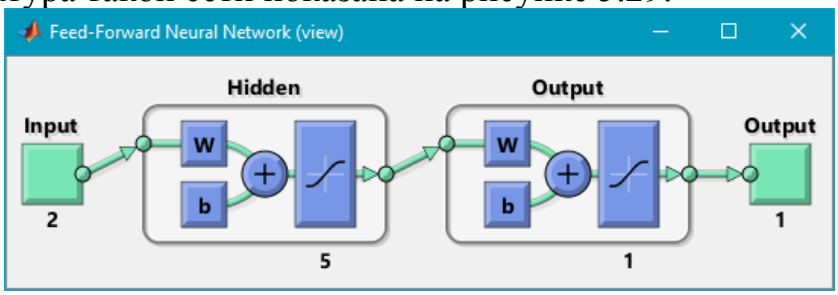


Рисунок 5.29 – Данная сеть тоже справится с задачей XOR

Однако, как видно из рисунка 5.30, ошибка обучения опустится в район 10^{-9}, что больше, чем у первой сети, что в целом хуже.

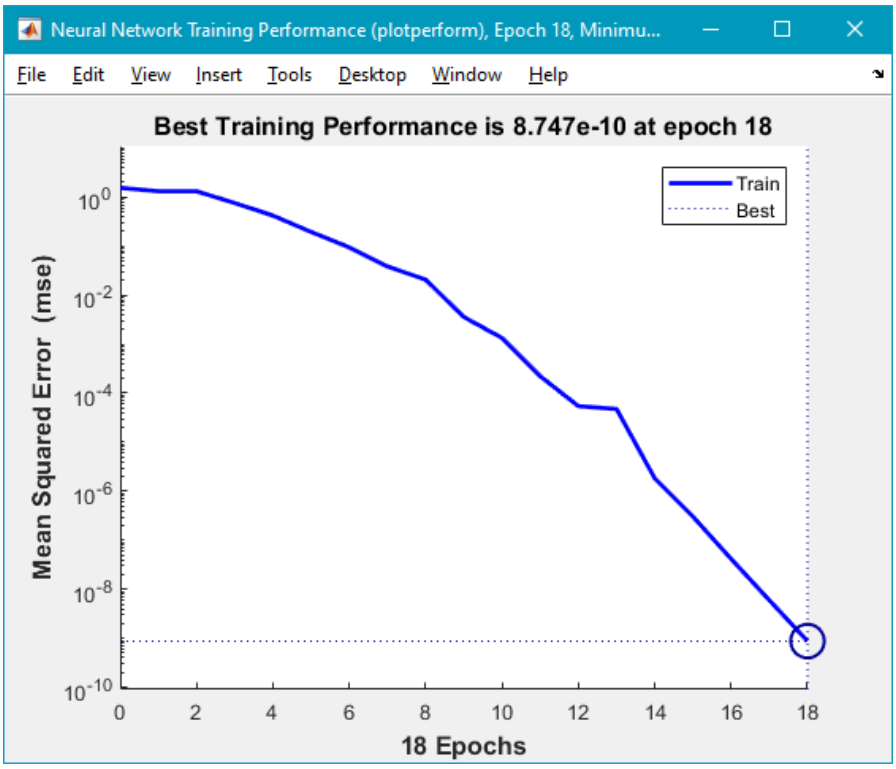


Рисунок 5.30 – Ошибка обучения для сети 2-5-1

Граница решений для этой сети показана на рисунке 5.31.

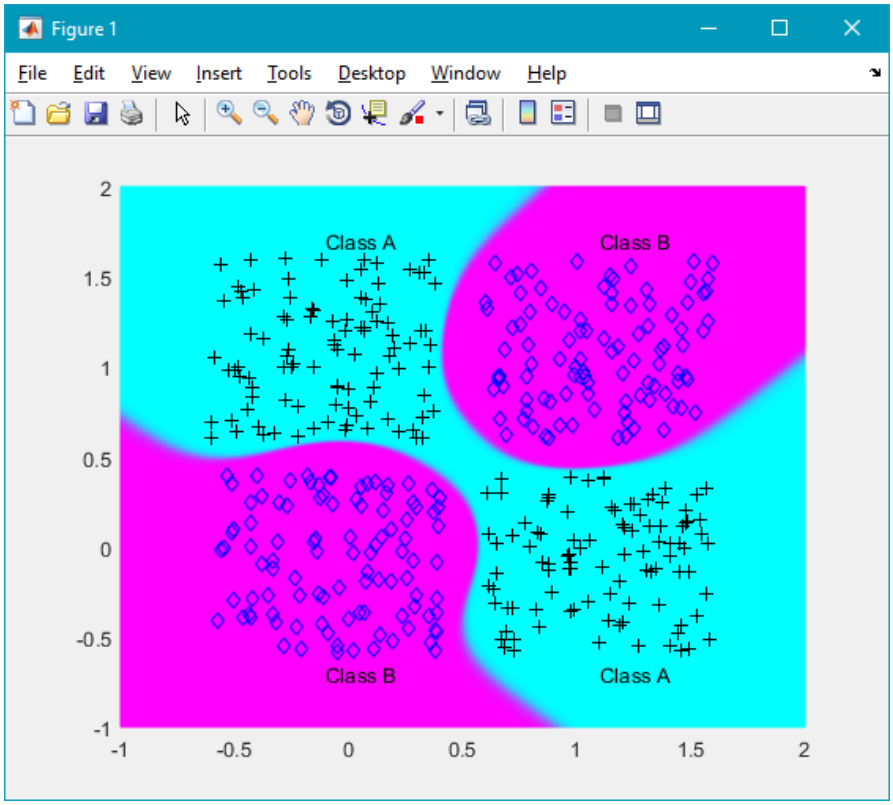


Рисунок 5. 31 – Граница решений для сети с пятью скрытыми нейронами

Сеть со структурой 2-3-1 даст ошибку, показанную на рисунке 5.32.

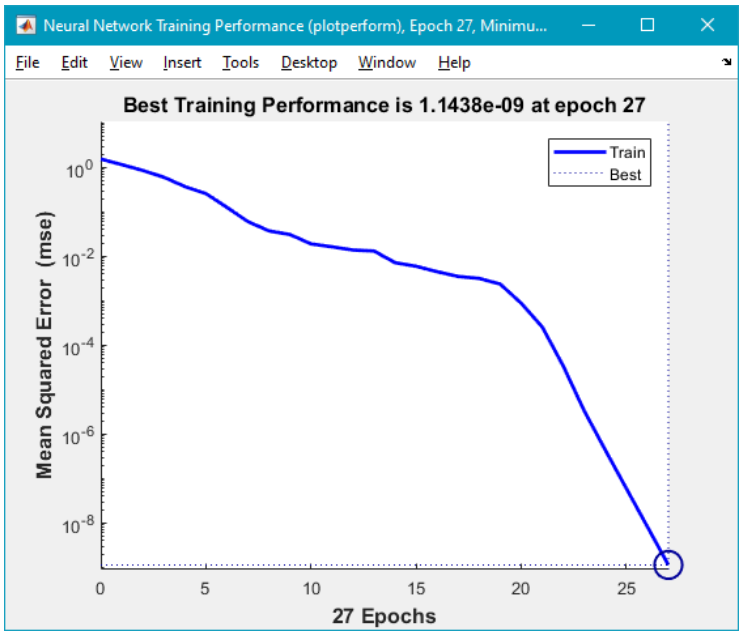


Рисунок 5.32 – Ошибка обучения для сети 2-3-1

Получившаяся граница решений для сети с тремя скрытыми нейронами показана на рисунке 5.33.

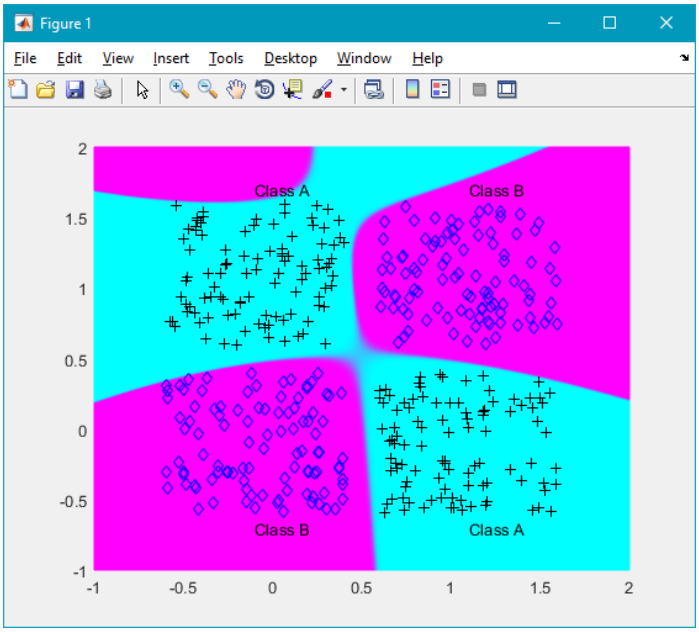


Рисунок 5.33 – Граница решений для сети из трёх нейронов в скрытом слое

Можно сделать вывод, что чем меньше количество нейронов в скрытом слое, тем грубее будут разделены классы и граница решений может в некоторых местах принимать хаотический вид (рисунок 5.33).

Конечно, такой разброс в решениях получается вследствие лёгкой разделимости классов.

Пример 7. Применение сети прямого распространения к разделению 4-х классов.

Теперь применим сеть прямого распространения к разделению 4-х классов (ранее решали эту задачу с помощью персептрона).

```
close all, clear all, clc, format compact
K = 100;
q = .6;
A = [rand(1, K)-q; rand(1, K)+q];
B = [rand(1, K)+q; rand(1, K)+q];
C = [rand(1, K)+q; rand(1, K)-q];
D = [rand(1, K)-q; rand(1, K)-q];
figure(1);
plot(A(1, :), A(2, :), 'k+');
hold on;
grid on;
plot(B(1, :), B(2, :), 'b*');
plot(C(1, :), C(2, :), 'kx');
plot(D(1, :), D(2, :), 'bd');
text(.5-q, .5+2*q, 'Class A');
text(.5+q, .5+2*q, 'Class B');
text(.5+q, .5-2*q, 'Class C');
text(.5-q, .5-2*q, 'Class D');
% Кодируем классы
a = [-1 -1 -1 +1]';
b = [-1 -1 +1 -1]';
c = [-1 +1 -1 -1]';
d = [+1 -1 -1 -1]';
P = [A B C D];
T = [repmat(a, 1, length(A)) repmat(b, 1, length(B)) repmat(c, 1, length(C)) repmat(d, 1, length(D))];
net =feedforwardnet([4 3]);
net.divideParam.trainRatio = 1;
net.divideParam.valRatio = 0;
net.divideParam.testRatio = 0;
[net, tr, Y, E] = train(net, P, T);
view(net);
% Получаем в “m” все максимальные элементы по столбцам, а в “i”
% их строковые индексы
[m, i] = max(T);
% Получаем в j индексы строк максимальных элементов
[m, j] = max(Y);
N = length(Y);
k = 0;
% Сравниваем эти индексы, они должны совпасть, если они не
% совпадают, то увеличиваем k на 1, т.е. k будет содержать количество
% ответов ИНС, которые не совпадают с метками
if find(i - j),
k = length(find(i-j));
end
% Высчитываем это количество в процентах
fprintf('Correct classified samples: %.if%% samples\n', 100*(N-k)/N);
figure;
% Строим верхний график на канве
subplot(211);
plot(T');
```

```
title('Targets');
ylim([-2 2]);
grid on;
% Строим нижний график на канве
subplot(212);
plot(Y');
title('Network response');
xlabel('# sample');
ylim([-2 2]);
grid on;
span = -1:.01:2;
[P1, P2] = meshgrid(span, span);
pp = [P1(:) P2(:)]';
aa = net(pp);
figure(1);
% Наносим на сетку все ответы первого нейрона для
% всей выборки, они должны занять ¼ от площади всех классов
m = mesh(P1, P2, reshape(aa(1, :), length(span), length(span))-5);
set(m, 'facecolor', [1 0.2 .7], 'linestyle', 'none');
hold on;
% Тоже самое делаем для других нейронов
m = mesh(P1, P2, reshape(aa(2, :), length(span), length(span))-5);
set(m, 'facecolor', [1 1.0 0.5], 'linestyle', 'none');
m = mesh(P1, P2, reshape(aa(3, :), length(span), length(span))-5);
set(m, 'facecolor', [.4 1.0 0.9], 'linestyle', 'none');
m = mesh(P1, P2, reshape(aa(4, :), length(span), length(span))-5);
set(m, 'facecolor', [.3 .4 0.5], 'linestyle', 'none');
view(2);
```

Создаём 4 класса, каждый представлен 100 паттернами, рисунок 5.34.

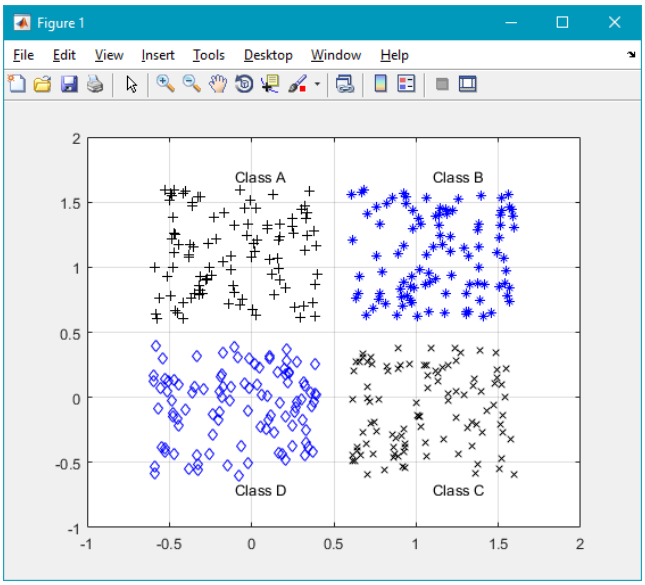


Рисунок 5.34 – Четыре класса, которые нужно разделить

Далее создаём сеть со структурой 2-4-3-4 и обучаем её, рисунки 3.35 – 3.37.

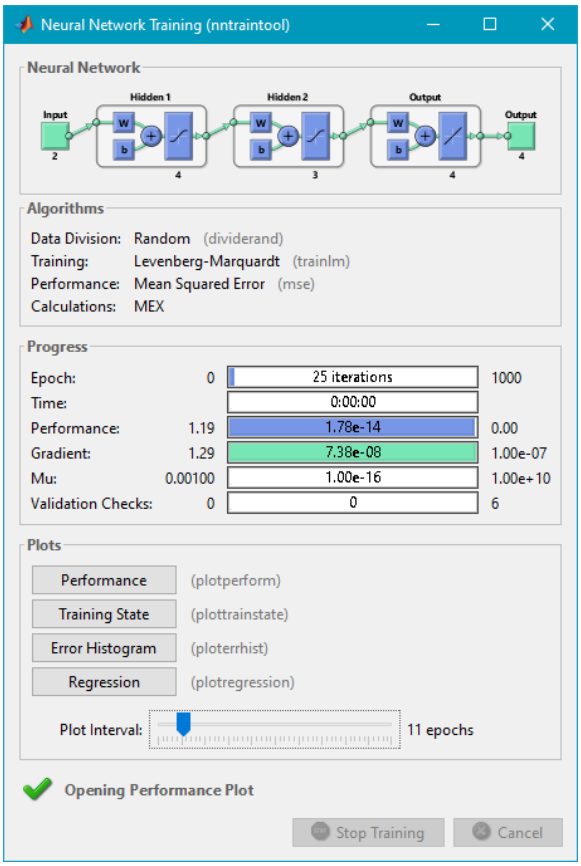


Рисунок 5.35 – Обучение сети, потребовалось 25 итерации

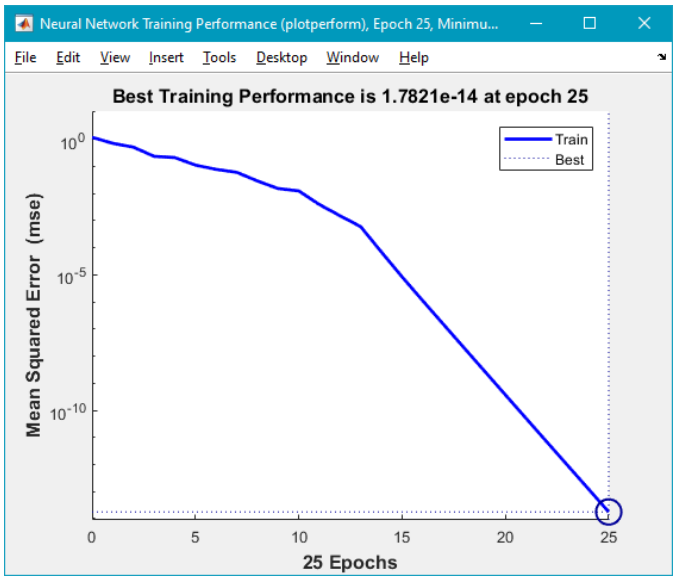


Рисунок 5.36 – Ошибка обучения очень хорошая, стала ниже 10^{-10} на четыре порядка

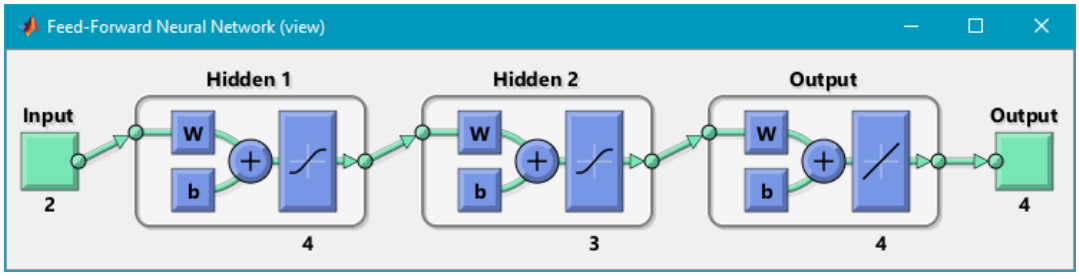


Рисунок 5.37 – Общая структура сети, которая использовалась

Разумеется, предложенная структура не единственная, на самом деле хватило бы сети с одним скрытым слоем и четырьмя нелинейными нейронами на выходном слое.

Метки и ответы сети изображены на рисунке 5.38, видно, что графики полностью совпадают, т.е. сеть точно моделирует все реакции на входные значения.

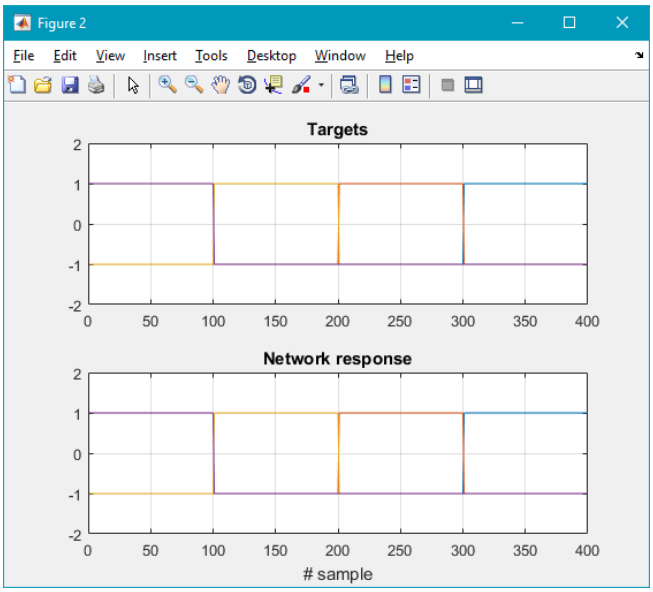


Рисунок 5.38 – График ожидаемых и реальных ответов ИНС

После построения границ решения можно видеть, что каждый класс отделён от другого, рисунок 5.39.

Если сравнить это решение с тем, которое было достигнуто с помощью персептрона, то станет очевидным, что возможности по отделению одних классов от других у ИНС прямого распространения значительно выше, чем у персептрона, т.к. такие сети позволяют строить более сложные границы.

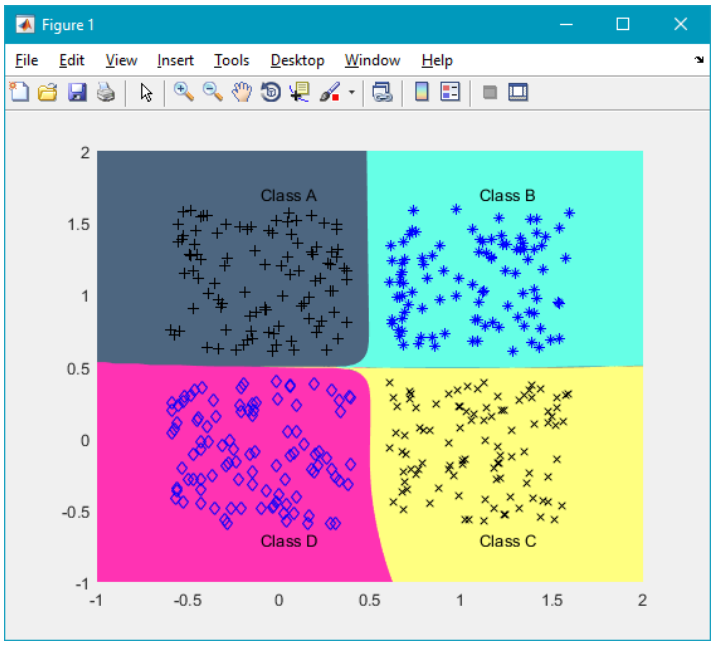


Рисунок 5.39 – Границы решения для задачи по разделению четырёх классов с помощью сети прямого распространения

Аппаратура и материалы. Для выполнения лабораторной работы необходимо использовать следующее: аппаратное обеспечение: персональный компьютер и программное обеспечение: операционную систему Windows 10 и выше; Microsoft Office13 и выше, системы компьютерной математики Machcad и MATLAB R2017a и выше.

Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

Выполните предложенные задания, предварительно рассмотрев и выполнив все приведенные примеры.

Задание 5.1. В индивидуальном варианте (таблица 5.1) дано расположение двух классов (A и B) по «углам» квадрата (по типу как на рисунке 5.39). Каждый «угол» представлен 200 точками. Требуется отделить класс A от B с помощью персептрона и полносвязанной многослойной сети прямого распространения.

Таблица 3.1 – Варианты заданий

Номер варианта	Условие задачи	Номер варианта	Условие задачи
1	$\begin{pmatrix} B & A \\ B & B \end{pmatrix}$	8	$\begin{pmatrix} B & A \\ B & A \end{pmatrix}$
2	$\begin{pmatrix} A & B \\ B & B \end{pmatrix}$	9	$\begin{pmatrix} A & B \\ B & A \end{pmatrix}$

Номер варианта	Условие задачи	Номер варианта	Условие задачи
3	$\begin{pmatrix} B & B \\ A & B \end{pmatrix}$	10	$\begin{pmatrix} B & A \\ A & B \end{pmatrix}$
4	$\begin{pmatrix} B & B \\ B & A \end{pmatrix}$	11	$\begin{pmatrix} A & A \\ B & A \end{pmatrix}$
5	$\begin{pmatrix} A & A \\ B & B \end{pmatrix}$	12	$\begin{pmatrix} B & A \\ A & A \end{pmatrix}$
6	$\begin{pmatrix} A & B \\ A & B \end{pmatrix}$	13	$\begin{pmatrix} A & B \\ A & A \end{pmatrix}$
7	$\begin{pmatrix} B & B \\ A & A \end{pmatrix}$	14	$\begin{pmatrix} A & A \\ A & B \end{pmatrix}$

Содержание отчета и его форма

Подготовьте отчет, в котором приведите технологию выполнения заданий.

Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) формулировку задания и технологию его выполнения;
4) ответы на контрольные вопросы;
5) приложение – файлы выполненных заданий.

Вопросы для защиты работы

1. Как кодируются классы для персептронов или сетей прямого распространения?
2. Опишите последовательность действий для нанесения входных значений на канву, чтобы визуально было видно расположение классов относительно друг друга.
3. Опишите последовательность действий для построения на канве границ решения задачи классификации.
4. Приведите пример сети прямого распространения с одним скрытым слоем, с помощью которой можно решить задачу о разделении 4-х классов в рассмотренной лабораторной работе.
5. Почему сеть прямого распространения может лучше справляться с задачами классификации, чем персептрон?

ЛАБОРАТОРНАЯ РАБОТА 6
Оптимизационные модели принятия решений. Постановка и решение задач оптимизации

Цель и содержание: Построение моделей задач и изучение методов их решения. Решение задачи принятия решений в условиях определенности.

Организационная форма занятий: решение проблемных задач, разбор конкретных ситуаций

Вопросы для обсуждения на лабораторном занятии: построение моделей оптимизационных задач принятия решений в условиях определенности и их решение современными средствами информационных технологий.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Изучите теоретический материал по данной теме.

Рассмотрим информационные модели принятия решений в условиях определенности на основе задач линейного программирования

Несмотря на требование линейности целевых функций (критериев оптимальности) и ограничений в линейном программировании решаются многочисленные задачи: распределения ресурсов; управления запасами; сетевого и календарного планирования; транспортные задачи, задачи теории игр и т.д.

Рассмотрим задачу на определение оптимального выпуска продукции. Имеются m видов ресурсов в количествах b_1, b_2, . , b_i, b_m и n видов изделий. Задана матрица $A=\|a_{ij}\|$, $i = 1,\ldots,m$, $j = 1,\ldots,n$, где a_{ij} характеризует нормы расхода i-го ресурса на единицу j-го вида изделий. Эффективность производства j-го вида изделий характеризуется показателем C_j, удовлетворяющим условию линейности. Нужно определить такой план выпуска изделий (оптимальный ассортимент), при котором суммарный показатель эффективности будет наибольший.

Обозначим количество единиц j-го вида изделий, выпускаемых предприятием, через x_j. Тогда математическая модель этой задачи будет иметь такой вид:

$$F(x)=\sum_{j=1}^{n} c_j x_j \quad (\max)$$

при ограничениях:

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad j=\overline{1,n} \qquad (6.1)$$

$$x_j \geq 0, \; j=\overline{1,n}.$$

Рассмотрим решение задачи линейного программирования в Excel.

Пример 1. Найти максимальное значение целевой функции:

$$F(x)=50x_1+40x_2$$

при ограничениях:

$$\begin{cases} 2x_1+5x_2\le 200, \\ 8x_1+5x_2\le 400, \\ \dots \end{cases}$$

$$x_1\ge 0,\ x_2\ge 0.$$

Решение. Для решения задачи необходимо выполнить следующие действия:

1. Заполнить рабочий лист следующим образом:

1.1. Ввести в ячейки (рисунок. 6.1):

A2 – формулу целевой функции, заменив x_1 и x_2 на C2 и C3: =E1*C2+F1*C3 или (=50*C1+40*C2). В первом случае в формуле указываются ссылки на ячейки, содержащие коэффициенты целевой функции, во втором случае просто вводятся их числовые значения;

A4 – формулу первого ограничения: =E3*C2+F3*C3 или (=2*C1+5*C2). Если ввести формулу, со ссылками на ячейки, указав какие из ячеек должны при копировании не меняться (для этого надо нажать на клавиатуре функциональную клавишу F4), то в ячейки A5 и A6 можно будет просто скопировать введенную формулу;

A5 – формулу второго ограничения: =E4*C2+F4*C3 или (=8*C1+5*C2);

A6 – формулу третьего ограничения: =E5*C2+F5*C3 или (=5*C2+6*C2);

A8 – ссылку на ячейку C2;

A9 – ссылку на ячейку C3.

1.2. Заполнить ячейки C2 и C3, положив начальные значения переменных равными нулю.

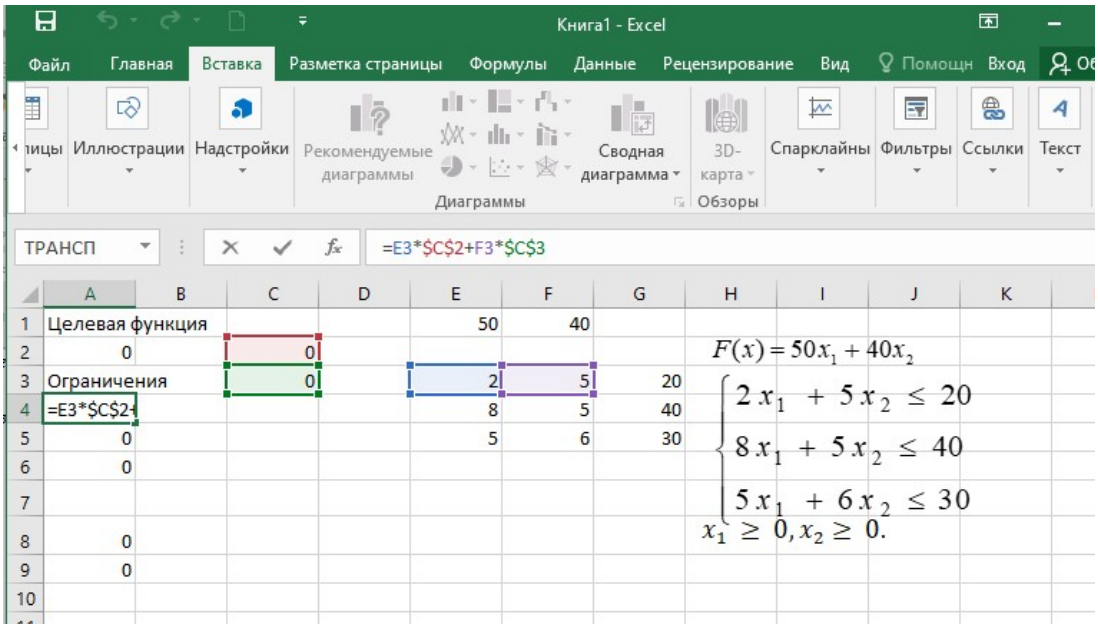


Рисунок 6.1 – Рабочий лист с условием задачи

2. Выделить ячейку, содержащую целевую функции, в нашем случае, это А2 и выбрать на вкладке **Данные**, в группе **Анализ**, надстройку **Поиск решений**. Появится окно диалога **Поиск решения** (рисунок 6.2), с помощью которого необходимо заполнить соответствующие поля:

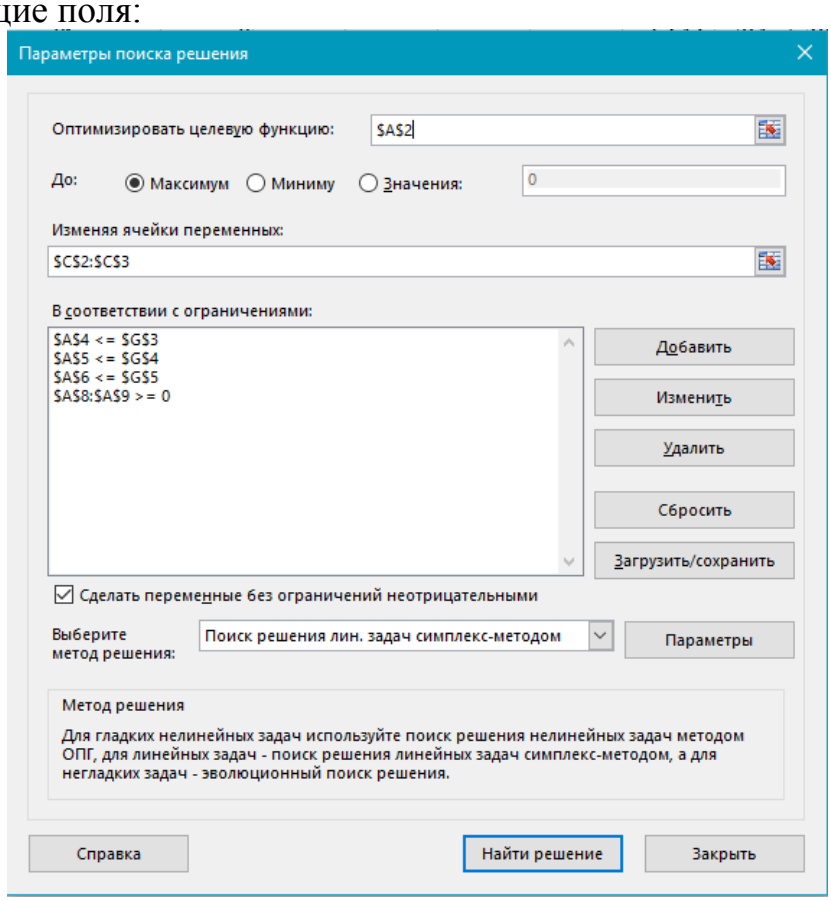


Рисунок 6.2 – Окно диалога **Поиск решения** с введенными данными

3. После окончания расчета Excel откроет окно диалога **Результаты поиска решения** (рисунок 6.3).

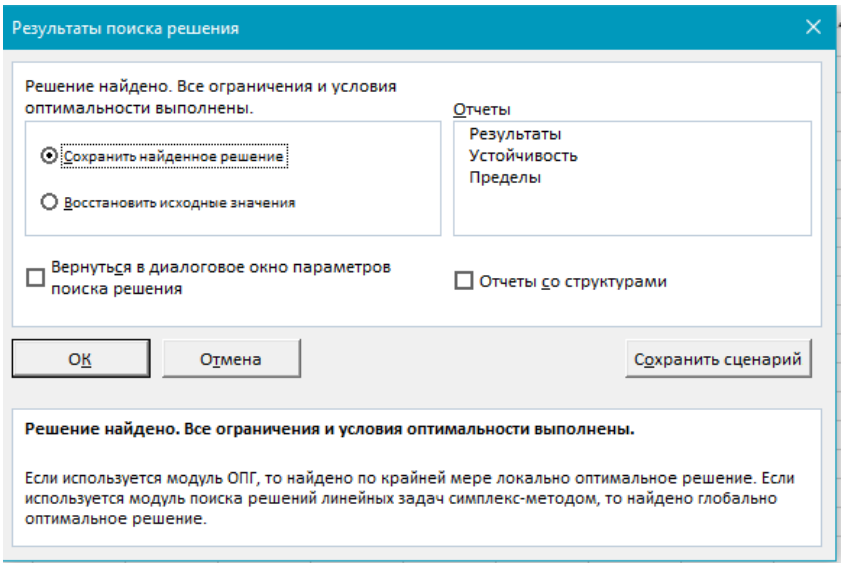


Рисунок 6.3 – Окно диалога **Результаты поиска решения**

Для просмотра результатов решения задачи необходимо нажать Enter или ОК. Получим максимальное значение функции равно 265,2173913, при значениях переменных $x_1 = 3{,}913043478$, $x_2 = 1{,}739130435$.

Понятие двойственности дает ощутимые практические результаты при построении алгоритмов решения задач линейного программирования. Для каждой задачи линейного программирования можно построить другую задачу, называемую двойственной.

Запишем обе задачи:

Прямая

$$F(x)=\sum_{j=1}^{n} c_j x_j (\min)$$

при ограничениях:

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, i \in M¿ \quad , \quad i=\overline{(1,k)}$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \quad , \quad i \in M¿ \quad , \quad i=\overline{(k+1,m)}$$

$$x_j \geq 0 \quad , \quad j=\overline{(1,l,)}$$

Двойственная

$$\phi(y)=\sum_{i=1}^{m} b_i y_i (\max)$$

при ограничениях:

$$\sum_{i=1}^{m} a_{ji} y_i \leq c_j, j \in J \quad , \quad j=\overline{(1,l)}$$

$$\sum_{i=1}^{m} a_{ji} y_i = c_j \quad , \quad j=\overline{(l+1,n)}$$

$$y_i \geq 0 \quad , \quad i=\overline{(i=1,k)}.$$

Симметричность обеих задач очевидна.

Неравенству первой задачи соответствует не отрицательность переменной во второй. Равенству одной задачи соответствует свободная переменная другой, задача двойственная к двойственной есть исходная (прямая) задача.

Таким образом:

Во взаимно-двойственных задачах всегда:

1. Одна из задач является задачей (*max*), а другая – задачей (*min*), в системе ограничений задачи (*max*) неравенств записаны со знаком ≤, а в

системе ограничений задачи (*min*) – со знаком ≥.

2. Каждому ограничению одной задачи соответствует переменная другой задачи: номер переменной совпадает с номером ограничения, при этом ограничению, записанному в виде неравенства, соответствует переменная, связанная условием неотрицательности.

3. Коэффициенты целевой функции одной задачи соответственно равны свободным членам системы ограничений другой задачи.

4. Матрица условий одной задачи получается из матрицы условий другой задачи с помощью транспонирования.

Взаимно-двойственными являются следующие задачи:

Прямая

$$F(x)=\sum_{j=1}^{n} c_j x_j \quad (\min)$$

при ограничениях:

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad j=\overline{1,n}$$

$$x_j \geq 0,\ j=\overline{1,n}.$$

Двойственная

$$\varphi(y)=\sum_{i=1}^{m} b_i y_i \quad (\max)$$

при ограничениях:

$$\sum_{i=1}^{m} a_{ji} y_i \leq c_j, \quad i=\overline{1,m}$$

$$y_i \geq 0,\ i=\overline{1,m}.$$

Канонический вид

Прямая

$$F(x)=\sum_{j=1}^{n} c_j x_j \quad (min)$$

при ограничениях:

$$\sum a_{ij} x_j - b_i,\ i=\overline{1,m}$$

$$x_j \geq 0,\ j=\overline{1,n}.$$

Двойственная

$$\varphi(y)=\sum_{i=1}^{m} b_i y_i \quad (max)$$

при ограничениях:

$$\sum_{i=1}^{m} a_{ji} y_i = c_j, \quad j=\overline{1,n}$$

$$y_i \geq 0,\ i=\overline{1,m}.$$

Пример 2. Рассмотрим пару двойственных задач.

Прямая задача. Найти:

$$F=5x_1-x_2+8x_3-x_4 \quad (\max)$$

при ограничениях:

$$\begin{cases} 2x_1+5x_2-x_3+7x_4=2, \\ x_1-x_2+5x_3-x_4 \leq 3. \end{cases}$$

$$x_1 \geq 0,\ x_3 \geq 0$$

Двойственная задача. Найти:

$$\phi=2y_1+3y_2+5y_3 \quad (\min)$$

при ограничениях:

$$\begin{cases} 2y_1+y_2+y_3 \geq 5, \\ 5y_1-y_2-y_3=-1, \\ y_1+5y_2+5y_3 \geq 8. \end{cases}$$

$$y_2 \geq 0$$

Запишем матрицы соответствующих задач: $A=\begin{vmatrix} 2 & 5 & -1 & 7 \\ 1 & -1 & 5 & -1 \\ 1 & -1 & 3 & 7 \end{vmatrix}$,

$A^T=\begin{vmatrix} 2 & 1 & 1 \\ 5 & -1 & -1 \\ -1 & 5 & 3 \\ 7 & -1 & 7 \end{vmatrix}$. $C = \begin{vmatrix} 5 & -1 & 8 \end{vmatrix}$; $C^T=\begin{vmatrix} 5 \\ -1 \\ 8 \end{vmatrix}$; $B=\begin{vmatrix} 2 \\ 3 \\ 5 \\ 3 \end{vmatrix}$; $B^T=\begin{vmatrix} 2 & 3 & 5 & 3 \end{vmatrix}$;

$A^T=\begin{vmatrix} 2 & -1 & 1 & 2 \\ 2 & 1 & 1 & 1 \\ -1 & 4 & -2 & -2 \end{vmatrix}$.

Наиболее часто встречаются следующие частные случаи взаимно-двойственных задач:

1) если $I = 0$, а $J=N=\{1,2,\ldots,n\}$, то имеем симметричную пару;

2) если $I=M=\{1,2,\ldots,m\}$, а $J=N=\{1,2,\ldots,n\}$, то имеем не симметричную пару.

Рассмотрим симметричную пару двойственных задач.

Прямая задача. Найти:

$$F(x)=\sum_{j=1}^{n} c_j x_j \quad (\max)$$

при ограничениях:

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad i=\overline{1,m}$$

$$x_j \ge 0, \ j=\overline{1,n}.$$

Двойственная задача. Найти:

$$\phi(y)=\sum_{i=1}^{m} b_i y_i \quad (\min)$$

при ограничениях:

$$\sum_{i=1}^{m} a_{ji} y_i \ge c_j, \quad j=\overline{1,n}$$

$$y_i \ge 0, \ i=\overline{1,m}.$$

Рассмотрим несимметричные двойственные задачи.

В несимметричных двойственных задачах система ограничений исходной задачи задается в виде неравенств, причем в последней переменные могут быть отрицательными.

Прямая задача. Найти:

$$F(x)=\sum_{j=1}^{n} c_j x_j \quad (\max)$$

при ограничениях:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i=\overline{1,m}$$

Двойственная задача. Найти:

$$\varphi(y)=\sum_{i=1}^{m} b_i y_i \quad (\min)$$

при ограничениях:

$$\sum_{i=1}^{m} a_{ji} y_i \ge c_j, \quad j=\overline{1,n}$$

$x_j \geq 0,\ j=\overline{1,n}.$ $y_i \geq 0,\ i=\overline{1,m}.$

Простейшие свойства взаимно однозначных двойственных задач:

- если x допустимое решение прямой задачи, а y допустимое решение двойственной задачи, то $F(x) \leq \phi(y)$;
- если x и y допустимые решения прямой задачи и двойственной задачи, и $F(x) = \phi(y)$, то x и y – оптимальные решения этих задач.

Свойства двойственных задач линейного программирования

Для взаимно-двойственных задач имеет место один из взаимоисключающих случаев, т.е. справедливо одно и только одно из следующих утверждений:

1. В прямой и двойственной задачах имеются оптимальные решения, причём значения целевых функций на оптимальных решениях совпадают, т.е. $\min f(x) = \max \phi(y)$.
2. В прямой задаче допустимо множество не пустое, а целевая функция на этом множестве не ограничена сверху. При этом у двойственной задачи будет пустое допустимое множество.
3. В двойственной задаче допустимо множество не пусто, а целевая функция на этом множестве не ограничена снизу. При этом у прямой задачи допустимое множество оказывается пустым.
4. Обе из рассматриваемых задач имеют пустые допустимые множества.

Первая (теорема двойственности):

Если из пары двойственных задач одна обладает оптимальным планом, то и другая имеет решение, причем для экстремальных решений линейных функций выполняется соотношение $\min F(x) = \max \phi(y)$, если линейная функция одной из задач не ограничена, то и другая не имеет решений.

Вторая теорема двойственности

Пусть $X=(x_1, x_2, \ldots, x_j, \ldots, x_n)$ – допустимое решение прямой задачи, а $Y=\{y_1, y_2, \ldots, y_i, \ldots, y_m\}$ – допустимое решение двойственной задачи. Для того, чтобы допустимые решения X и Y прямой двойственной задачи были оптимальными необходимо и достаточно, чтобы выполнялись следующие соотношения:

а) $Y(A \cdot X - b) = 0$ иначе $y_i\left(\sum_{j=1}^{n} a_{ij} \cdot x_j - b_i\right) = 0$

б) $(C - Y \cdot A)X = 0$ $x_j\left(-\sum_{i=1}^{m} a_{ji} \cdot y_y + c_j\right) = 0$

Условия:

а) равносильно следующему:

если $y_i > 0$, то $\sum_{j=1}^{n} a_{ij} \cdot x_j = b_i, i = 1, \ldots, m$;

если $y_i = 0$, то $\sum_{j=1}^{n} a_{ij} \cdot x_j > b_i$;

б) равносильно следующему:

если $x_j > 0$, то $c_j = \sum_{i=1}^{m} a_{ji} \cdot y_i, j = 1, \ldots, n$;

если $x_j = 0$, то $c_j > \sum_{i=1}^{m} a_{ji} \cdot y_i$.

Может случиться, что одновременно $y_i = 0$ и $\sum_{j=1}^{n} a_{ij} \cdot x_j = b_i$. Но всегда существует по крайней мере одна пара.

Запишем двойственные задачи в векторном виде.

Несимметричные двойственные задачи

Прямая. Найти матрицу столбец $X = |x_1 \quad x_2 \quad \ldots \quad x_i \quad \ldots \quad x_n|$, которая удовлетворяет ограничениям $A \cdot X = B$, $x \geq 0$ и минимизирует функцию $F = C \cdot X$.

Двойственная. Найти матрицу строку $Y = |y_1 \quad y_2 \quad \ldots \quad y_i \quad \ldots \quad y_m|$, которая удовлетворяет ограничениям $Y \cdot A \leq C$ и максимизирует линейную функцию $\varphi = Y \cdot B$.

В общих задачах: $C = |c_1 \quad c_2 \quad \ldots \quad c_j \quad \ldots \quad c_n|$ – матрица строка, представляющая собой коэффициенты при неизвестных в целевой функции; $B = |b_1 \quad b_2 \quad \ldots \quad b_i \quad \ldots \quad b_m|$ – матрица столбец свободных членов в системе ограничений; $A = |a_{ij}|$ – матрица коэффициентов при неизвестных в системе ограничений.

***Пример* 3**. Найти $F = x_2 - x_4 - 3x_5 (\min)$ при ограничениях:

$$\begin{cases} x_1 + 2x_2 - x_4 + x_5 = 1 \\ -4x_2 + x_3 + 2x_4 - x_5 = 2 \end{cases}$$

Матрица строка $C = |0 \quad 1 \quad 0 \quad -1 \quad -3 \quad 0|$ – коэффициенты при

$$B=\begin{vmatrix}1\\2\\5\end{vmatrix}$$

неизвестных в целевой функции. Матрица столбец – свободные

$$A=\begin{vmatrix}1 & 2 & 0 & -1 & 1 & 0\\0 & -4 & 1 & 2 & -1 & 0\\0 & 3 & 0 & 0 & 1 & 1\end{vmatrix}$$

члены в системе ограничений, A = – матрица коэффициентов при неизвестных в системе ограничений.

Пусть $X=\begin{vmatrix}x_1 & x_2 & \dots & x_i & \dots & x_n\end{vmatrix}$ – допустимое решение прямой задачи. Вектор X является оптимальным решением этой задачи тогда и только тогда, когда среди решений системы уравнений $\sum_{i=1}^{m} a_{ji}\cdot y_i - c_j = 0$, при $x_j \neq 0$, $y_i = 0$, при $\sum_{j=1}^{n} a_{ij}\cdot x_j < b_i$ содержится хотя бы одно допустимое решение двойственной задачи.

***Пример* 4.** Вектор X = (3; 0,1; –3) – допустимое решение задачи $F=-2x_1-x_2+x_3+x_4 \,(\max)$ при ограничениях:

$$\begin{cases} x_1 - x_2 + 2x_3 = 2, \\ 3x_2 - 7x_3 + 3x_4 = 2, \\ \dots \end{cases}$$

В данном случае соотношений $\sum_{j=1}^{n} a_{ij}\cdot x_j < b_i$.

Симметричные двойственные задачи

Прямая (исходная):

Найти матрицу-столбец $X=\begin{vmatrix}x_1 & x_2 & \dots & x_i & \dots & x_n\end{vmatrix}$, которая удовлетворяет системе ограничений: $A\cdot X \geq B$; $X \geq 0$ и минимизирует функцию $F = C\cdot X$.

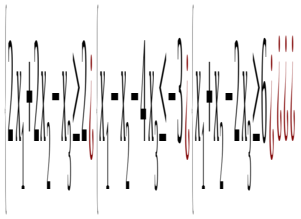
Двойственная задача:

Найти матрицу-строку $Y=\begin{vmatrix}y_1 & y_2 & \dots & y_i & \dots & y_m\end{vmatrix}$, которая удовлетворяет системе ограничений: $Y\cdot A \leq C$; $Y \geq 0$ и максимизирует линейную функцию $\varphi = Y\cdot B$.

***Пример* 5.** Найти:

$$F = x_1+2x_2+3x_3 \quad (\min)$$

при ограничениях:



$x_j>0$ (j = 1, 2, 3).

Чтобы привести систему ограничений к необходимому виду ($A \cdot X \geq B$), необходимо второе неравенство умножить на (–1), после чего запишем соответствующие матрицы:

$$C = \begin{vmatrix} 1 & 2 & 3 \end{vmatrix}, B = \begin{vmatrix} 2 \\ 3 \\ 6 \end{vmatrix}; \quad A = \begin{vmatrix} 2 & 2 & -1 \\ -1 & 1 & 4 \\ 1 & 1 & -2 \end{vmatrix}; \quad C^T = \begin{vmatrix} 1 \\ 2 \\ 3 \end{vmatrix}; \quad B^T = \begin{vmatrix} 2 & 3 & 5 & 3 \end{vmatrix};$$

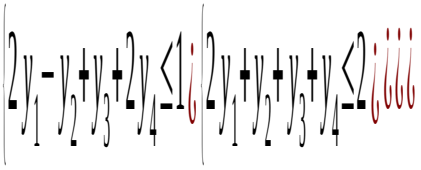
$$A^T = \begin{vmatrix} 2 & -1 & 1 & 2 \\ 2 & 1 & 1 & 1 \\ -1 & 4 & -2 & -2 \end{vmatrix}.$$

В результате двойственная задача имеет вид:

Найти:

$$\varphi = 2y_1+3y_2+6y_3+3y_4 \,(\max)$$

при ограничениях:



$y_j \geq 0$, (j=1, 2, 3, 4).

Аппаратура и материалы. Для выполнения лабораторной работы необходимо использовать следующее: аппаратное обеспечение: персональный компьютер и программное обеспечение: операционную систему Windows 10 и выше; Microsoft Office13 и выше, системы компьютерной математики Machcad и MATLAB R2017a и выше.

Указания по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

Выполните предложенные задания, предварительно решив все приведенные в данной работе примеры.

Задание 6.1. Постановка задачи линейного программирования.

Построить математическую модель планирования производства. Для

изготовления трех видов продукции P_1, P_2, P_3 используют три вида сырья S_1, S_2, S_3. Запасы сырья, число единиц сырья, затрачиваемых на изготовление единицы продукции, прибыль, получаемая от единицы продукции P_1, P_2, P_3 – соответственно приведены в таблице 6.1 (цифры условные).

Таблица 6.1 – Запасы сырья S_1, S_2, S_3, число единиц сырья, затрачиваемых на изготовление единицы продукции, прибыль, получаемая от единицы продукции P_1, P_2, P_3.

Вид сырья	Число единиц сырья, затрачиваемых на изготовление единицы продукции			Запасы сырья
	P_1	P_2	P_3	
S_1	4	2	1	150 000
S_2	6	0	2	170 000
S_3	0	2	4	100 000
Прибыль от ед. прод.	100	150	200	

Необходимо составить такой план производства продукции, при котором прибыль от ее реализации будет максимальной.

- Записать задачу в каноническом виде и решить её в Excel, используя инструмент **Поиск решения**.

Задание 6.2. Индивидуальное. Для исходной задачи записать двойственную задачу, решить обе задачи с помощью инструментального средства **Поиск решения** и проанализировать найденные результаты, используя теоремы двойственности.

1. $f = 4x_1 + 5x_2 + 6x_3 \prime$ (min)

при ограничениях:

$$\begin{cases} x_1 + x_2 + x_3 \geq 5 \\ x_1 - x_2 + 2x_3 \geq 1 \\ x_1 - x_2 - 4x_3 \leq -3 \\ x_1 - x_2 + 8x_3 \geq 4 \end{cases}$$

$$x_j \geq 0, \prime \left(j = \overline{1,3} \right).$$

2. $f = 2x_1 - 2x_2 + 3x_3 + 4x_4$ (max)

при ограничениях:

$$\begin{cases} x_1 + 2x_2 + x_3 + x_4 \leq 2 \\ 2x_1 - x_2 + 2x_3 - 3x_4 \geq 3 \\ 3x_1 + 4x_2 - 5x_3 + 2x_4 \leq 4 \end{cases}$$

$$x_j \geq 0, \prime \left(j = \overline{1,4} \right).$$

3. $f = 5x_1 - x_2 - 4x_3 \prime (\max)$

при ограничениях:

$$\begin{cases} -x_2 + 2x_3 \geq 9 \\ -x_1 + x_2 \geq 1 \\ x_1 + x_2 - 3x_3 \geq 8 \\ x_1 - x_3 \leq 4 \end{cases}$$

$$x_j \geq 0, \prime \left(j = \overline{1,3} \right).$$

4. $f = 2x_1 + 3x_3 + \frac{5}{2} x_3 \prime (\min)$

при ограничениях:

$$\begin{cases} 2x_1 + x_2 + 3x_3 \geq 6 \\ 2x_1 + 4x_2 + 3x_3 \geq 16 \\ 3x_1 + 4x_2 + 2x_3 \geq 12 \end{cases}$$

$$x_j \geq 0, \prime \left(j = \overline{1,3} \right).$$

5. $f=4x_1+6x_2+3x_3$ (min)

при ограничениях:

$$\begin{cases}3x_1+x_2+2x_3\geq 9\\ x_1+2x_2+3x_3\geq 8\\ x_1+6x_2\geq 12\end{cases}$$

$$x_j\geq 0,\ j=\overline{1,3}.$$

6. $f=2x_1+3x_2+7x_3$ (max)

при ограничениях:

$$\begin{cases}2x_1+3x_2+x_3\leq 9\\ -x_1-2x_2-3x_3\geq -8\\ 5x_1+6x_2\leq 12\end{cases}$$

$$x_j\geq 0,\ j=\overline{1,3}.$$

7. $f=6x_1+8x_2+x_3$ (max)

при ограничениях:

$$\begin{cases}x_1+x_2+x_3\leq 3\\ x_1+2x_2\leq 4\end{cases}$$

$$x_j\geq 0,\ j=\overline{1,3}.$$

8. $f=27x_1+10x_2+15x_3+28x_4$ (max)

при ограничениях:

$$\begin{cases}3x_1+2x_2+x_3+2x_4\leq 2\\ 3x_1+x_2+3x_3+4x_4\leq 5\end{cases}$$

$$x_j\geq 0,\ j=\overline{1,4}.$$

9. $f=3x_1+2x_2-6x_3$ (max)

при ограничениях:

$$\begin{cases}2x_1-3x_2+x_3\leq 18\\ -3x_1+2x_2-2x_3\leq 24\\ x_1+3x_2-4x_3\leq 36\end{cases}$$

$$x_j\geq 0,\ j=\overline{1,3}.$$

10. $f=3x_1-7x_2-4x_3$ (max)

при ограничениях:

$$\begin{cases}-2x_1-3x_2-2x_3\leq 12\\ -4x_1-4x_2-3x_3\leq 24\\ 5x_1+5x_2+3x_3\leq 15\end{cases}$$

$$x_j\geq 0,\ j=\overline{1,3}.$$

11. $f=x_1+3x_2-5x_4$ (max)

при ограничениях:

$$\begin{cases}2x_1+4x_2+x_3+2x_4=28\\ -3x_1+5x_2-3x_4\leq 30\\ 4x_1-2x_2+?8x_4\leq 32\end{cases}$$

$$x_j\geq 0,\ j=\overline{1,4}.$$

12 $f=3x_1+2x_5-5x_6$ (max)

при ограничениях:

$$\begin{cases}2x_1+x_2-3x_5+5x_6=30\\ 4x_1+x_3+2x_5-4x_6=28\\ -3x_1+x_4-3x_5+6x_6=24\end{cases}$$

$$x_j\geq 0,\ j=\overline{1,6}.$$

13. $f=x_1+x_2-3x_3$ (max)

при ограничениях:

$$\begin{cases}-x_2-2x_3\leq -7\\ -x_1+3x_2\geq 10\\ x_1+x_2-2x_3\geq 4\\ x_1+x_2-x_3\leq 4\end{cases}$$

$$x_j\geq 0,\ j=\overline{1,3}.$$

14. $f=2x_1-3x_3+4x_3$ (min)

при ограничениях:

$$\begin{cases}x_1+2x_2+3x_3\geq 5\\ 2x_1+3x_2-4x_3\geq 10\\ 3x_1+x_2+-7x_3\geq 13\end{cases}$$

$$x_j\geq 0,\ j=\overline{1,3}.$$

15. $f=4x_1-5x_2+2x_3$ (min)

16. $f=x_1+4x_2-7x_3$ (min)

при ограничениях:

при ограничениях:

$$\begin{cases} 5x_1-3x_2+4x_3\ge 9 \\ x_1-4x_2+3x_3\ge 8 \\ x_1+x_2\ge 10 \end{cases}$$

$$x_j\ge 0,\ j=\overline{1,3}.$$

17.

$$f=3x_1-7x_2+4x_3\ (\min)$$

при ограничениях:

$$\begin{cases} x_1-8x_2+3x_3\ge 5 \\ 3x_1+4x_2-7x_3\ge 18 \\ x_1+7x_2\ge 12 \end{cases}$$

$$x_j\ge 0,\ j=\overline{1,3}.$$

18. $f=-6x_1+2x_2-9x_3$ (max)

при ограничениях:

$$\begin{cases} 7x_1-5x_2+3x_3\le 13 \\ -9x_1+3x_2-5x_3\ge -8 \\ x_1-4x_2\le 13 \end{cases}$$

$$x_j\ge 0,\ j=\overline{1,3}.$$

19. $f=-4x_1+x_2-3x_3$ (max)

при ограничениях:

$$\begin{cases} 6x_1+3x_2-4x_3\ge 15 \\ 2x_1+4x_2-3x_3\ge 18 \\ x_1+7x_2\ge 1 \end{cases}$$

$$x_j\ge 0,\ j=\overline{1,3}.$$

20. $f=11x_1+4x_2-7x_3$ (min)

при ограничениях:

$$\begin{cases} 2x_1-3x_2+9x_3\le 3 \\ -4x_1-9x_2-6x_3\ge 7 \\ 5x_1-8x_2\le 1 \end{cases}$$

$$x_j\ge 0,\ j=\overline{1,3}.$$

Задание 4. Для двойственной симметричной задачи записать исходную и найти ее решение:

$$j=2y_1+4x_2+12y_4\ (\min)$$

при ограничениях:

$$\begin{cases} y_1+2y_2+y_3+4y_4\ge 10 \\ 2y_1+y_2-2y_3+3y_4\ge 4 \end{cases}$$

$$y_j\ge 0,\ (j=\overline{1,4}).$$

Содержание отчета и его форма

Подготовьте отчет, в котором опишите технологию решения задач линейного программирования средствами Excel, используя задания своего варианта. Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) условия выполненных заданий и их решение;
4) технологию решения задач с помощью прикладного программного обеспечения, выводы;
5) ответы на контрольные вопросы.

Вопросы для защиты работы

1. Дайте определение математического программирования.

2. Что называется математической моделью оптимизационной задачи?
3. Как строится математическая модель?
4. Как перейти от неравенств к уравнениям?
5. Какие переменные называются дополнительными (слабыми)?
6. Расскажите известные вам методы непосредственного решения простейших оптимизационных задач.
7. В чем заключается сущность двойственности в линейном программировании?
8. Сформулируйте 1-ю и 2-ю теорему двойственности.
9. Какие задачи линейного программирования относятся к симметричным и несимметричным? В чем их отличие?
10. Как по решению исходной задачи найти решение двойственной и наоборот?

ЛАБОРАТОРНАЯ РАБОТА 7
Построение и анализ моделей задач транспортного типа, решение задач методом потенциалов

Цель и содержание: Научиться строить и анализировать модели транспортной задачи.

Организационная форма занятий: решение проблемных задач, разбор конкретных ситуаций

Вопросы для обсуждения на лабораторном занятии: построение, решение и анализ модели транспортной задачи.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Изучите теоретический материал по данной теме, используя материал изложенный ниже.

Классические транспортные задачи относятся к области линейного программирования, и математический аппарат решения этих задач хорошо отработан и известен. Сфера применения моделей задач транспортного типа обширна. Сюда можно отнести проблемы, например, связанные с планированием производства и перевозок, с созданием вычислительных и информационных систем, с распределениями ресурсов, запасов и так далее, то есть в формальных терминах транспортной задачи формируется большое число задач, отнюдь не связанных с перевозками продуктов.

Сформулируем математическую модель транспортной задачи.

Некоторый однородный продукт, сосредоточенный у m поставщиков A в количестве a_i (i = 1, 2, ..., m) единиц соответственно, необходимо доставить n потребителям B в количестве b_j (j = 1, 2, ..., n) единиц. Известна стоимость C_{ij} перевозки единицы груза от i-того поставщика к j-тому потребителю. Необходимо составить план перевозок, позволяющий вывезти все грузы, полностью удовлетворить потребности и имеющий минимальную стоимость.

Обозначим через x_{ij} количество единиц груза, запланированных к перевозке от i-того поставщика к j-тому потребителю; тогда условие задачи можно записать в виде таблицы 4.1, которую в дальнейшем называют *матрицей планирования.*

Таблица 7.1 – Матрица планирования

Поставщики	Потребители, B_j				Запасы
A_i	B_1	B_2	…	B_n	a_i
A_1	c_{11} x_{11}	c_{12} x_{12}	…	c_{1n} x_{1n}	a_1
A_2	c_{21} x_{21}	c_{22} x_{22}	…	c_{2n} x_{2n}	a_2
…	…	…	…	…	...
A_m	c_{m1} x_{m1}	c_{m2} x_{m2}	…	c_{mn} x_{mn}	a_m

Потребности, b_j	b_1	b_2	…	b_n	$\Sigma a_i = \Sigma b_j$

Составим математическую модель задачи. Так как от i-го поставщика к j-тому потребителю запланировано к перевозке x_{ij} единиц груза, то стоимость перевозки составит $c_{ij}x_{ij}$. Стоимость всего плана выразится двойной суммой:

$$Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}.$$

Систему ограничений получим из следующих условий задачи:

а) все грузы должны быть вывезены, то есть,

$$\sum_{j=1}^{n} x_{ij} = a_i \quad , (i = 1, 2, ..., m),$$

эти уравнения получаются из строк таблицы 7.1;

б) все потребности должны быть удовлетворены, то есть,

$$\sum_{i=1}^{m} x_{ij} = b_j \quad , (j = 1,2, ..., n),$$

уравнения получаются из столбцов таблицы 4.1.

Таким образом, математическая модель транспортной задачи имеет следующий вид. Найти наименьшее значение функции:

$$Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad (7.1)$$

при ограничениях:

$$\begin{cases} \sum_{i=1}^{m} x_{ij} = b_j, j = \overline{1,n} \\ \end{cases} \qquad (7.2)$$

В рассмотренной модели предполагается, что суммарные запасы равны суммарным потребностям, то есть

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \quad .$$

Такая модель называется *закрытой*.

Рассмотрим пример построения математической модели транспортной задачи, методы построения опорного плана и метод решения задачи.

Пример. Четыре предприятия могут производить некоторую однородную продукцию в количестве соответственно 100, 250, 200 и 300. На эту продукцию есть заказ от пяти потребителей соответственно в

количестве 200, 200, 100, 100 и 250. Затраты с производством и доставкой единицы продукции задаются матрицей: $C = \begin{vmatrix} 10 & 7 & 4 & 1 & 4 \\ 2 & 7 & 10 & 6 & 11 \\ 8 & 5 & 3 & 2 & 2 \\ 11 & 8 & 12 & 16 & 13 \end{vmatrix}$.

Необходимо: построить математическую модель заданной транспортной задачи; найти опорный план методами: северо-западного угла; минимальной стоимости; двойного предпочтения; построить систему потенциалов и проверить найденные планы на оптимальность; найти оптимальный план методом потенциалов, взяв за первоначальный любой из опорных планов, не являющийся оптимальным.

Решение. *Построение математической модели заданной экономической системы.* Сформулируем математическую задачу: пусть некоторый однородный продукт, сосредоточен у $m = 4$ поставщиков A_i в количестве a_i ($i = 1, 2, ..., m$) единиц соответственно, необходимо доставить этот продукт n потребителям B_j в количестве b_j ($j = 1, 2, ..., n$) единиц. Известна стоимость c_{ij} перевозки единицы продукта от i-го поставщика к j-тому потребителю заданная матрицей стоимостей. Необходимо составить такой план перевозки продукта, который имеет минимальную стоимость.

Прежде чем составлять математическую модель заданной системы необходимо проверить условие закрытости задачи: а именно, что суммарные запасы равны суммарным потребностям, т. е. все запасы должны быть вывезены, а потребности удовлетворены:

$$\sum_{i=1}^{m=4} a_i = \sum_{j=1}^{n=5} b_j$$

$$\sum_{i=1}^{m=4} a_i = 100 + 250 + 200 + 300 = 850$$

$$\sum_{j=1}^{n=5} b_j = 200 + 200 + 100 + 100 + 250 = 850 .$$

Так как суммарные потребности равны суммарным запасам, имеем закрытую транспортную задачу. Обозначим через x_{ij} количество единиц груза, запланированных к перевозке от i-го поставщика к j-тому потребителю, и запишем условие задачи в виде матрицы планирования (таблица 7.2).

Таблица 7.2 – Матрица планирования для задачи, представленной в примере

Поставщики, A_i	Потребители, B_j					Запасы
	B_1	B_2	B_3	B_4	B_5	a_i
A_1	10 x_{11}	7 x_{12}	4 x_{13}	1 x_{14}	4 x_{15}	100
A_2	2	27	10	6	11	250

	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	
A_3	8 x_{31}	5 x_{32}	3 x_{33}	2 x_{34}	2 x_{35}	200
A_4	11 x_{41}	8 x_{42}	12 x_{43}	16 x_{44}	13 x_{45}	300
Потребности b_j	200	200	100	100	250	$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$

Составим математическую модель задачи. Так как от i-го поставщика к j-тому потребителю запланировано к перевозке x_{ij} единиц груза, то стоимость перевозки составит $c_{ij}x_{ij}$. Стоимость всего плана выразится двойной суммой:

$$F = \sum_{i=1}^{m=4} \sum_{j=1}^{n=5} c_{ij} x_{ij}.$$

Систему ограничений получим из следующих условий задачи:

а) все продукты должны быть вывезены, то есть,

$$\sum_{j=1}^{n=5} x_{ij} = a_i \quad , (i = 1, 2, 3, 4),$$

эти уравнения получаются из строк таблицы 4.2;

б) все потребности должны быть удовлетворены, то есть,

$$\sum_{i=1}^{m=4} x_{ij} = b_j \quad , (j = 1,2, ..., 5),$$

уравнения получаются из столбцов таблицы 7.2.

Таким образом, математическая модель данной задачи имеет следующий вид:

$$F = \sum_{i=1}^{m=4} \sum_{j=1}^{n=5} c_{ij} x_{ij} (\min) \qquad (7.3)$$

при ограничениях:

$$\begin{cases} \sum_{i=1}^{m=4} x_{ij} = b_j, j = \overline{1,5} \\ \end{cases} \qquad (7.4)$$

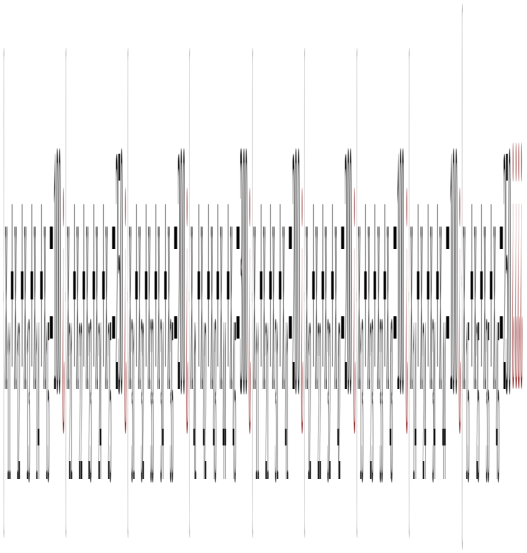
$$c_{ij}=\begin{vmatrix} 10 & 7 & 4 & 1 & 4 \\ 2 & 7 & 10 & 6 & 11 \\ 8 & 5 & 3 & 2 & 2 \\ 11 & 8 & 12 & 16 & 13 \end{vmatrix}, \qquad a_i=\begin{pmatrix} 100 \\ 250 \\ 200 \\ 300 \end{pmatrix},$$

где $b_j=(200,\ 200,\ 100,\ 100,\ 250)$.

Иначе математическую модель можно записать в следующем виде:

$$F=10x_{11}+7x_{12}+4x_{13}+1x_{14}+4x_{15}+2x_{21}+7x_{22}+10x_{23}+6x_{24}+11x_{25}+8x_{31}+5x_{32}+3x_{33}+2x_{34}+2x_{35}+11x_{41}+8x_{42}+12x_{43}+16x_{44}+13x_{45}\rightarrow \min$$

при ограничениях:



Таким образом, математическая постановка данной задачи состоит в нахождении такого не отрицательного решения системы линейных уравнений, при котором целевая функция принимает минимальное значение.

Определение опорного плана методами: северо-западного угла; минимальной стоимости; двойного предпочтения. Найдем опорный план методом северо-западного угла. Для этого запишем условие задачи в виде таблицы 7.3. Не учитывая стоимости перевозки единицы груза, начинаем удовлетворение потребностей первого потребителя B_1 за счет запаса поставщика A_1.

Для этого сравниваем a_1=100 с b_1=200, $a_1<b_1$, меньший из объемов, т. е. =100ед. записываем в левый нижний угол клетки A_1B_1. Запасы первого поставщика полностью израсходованы, поэтому остальные клетки первой строки вычеркиваем. Потребности B_1 остались неудовлетворенными на 200 –100 = 100 ед. Сравниваем этот остаток с запасами поставщика A_2: т. к.

100 < 250, то 100 ед. записываем в клетку A_2B_1, чем полностью

удовлетворяем потребности потребителя B_1, оставшиеся клетки в первом столбце вычеркиваем.

У поставщика A_2 осталось 150 ед. груза. Удовлетворяем потребности потребителя B_2 за счет оставшихся у поставщика A_2 запасов. Для этого сравниваем этот остаток с потребностями потребителя B_2: 150<200, записываем 150 ед. в клетку A_2B_2 и, т. к. запасы A_2 полностью израсходованы, прочеркиваем остальные клетки второй строки. Потребности B_2 остались неудовлетворенными на 50ед. Удовлетворяем их за счет поставщика A_3 и переходим к удовлетворению B_3 за счет остатка, имеющегося у поставщика A_3, и т. д. Процесс продолжаем до тех пор, пока не удовлетворим всех потребителей за счет запасов поставщиков. На этом построение первоначального опорного план заканчивается.

Таким образом, в таблице 7.3 в правых верхних углах клеток стоят числа, определяющие стоимость перевозки единицы грузов, в левых нижних углах – числа, определяющие план перевозок. Проверим, является ли план, построенный в таблице опорным. Видим, что, начиная движение от занятой клетки A_1B_1, вернуться не только в нее, но и в любую другую занятую клетку, двигаясь только по занятым клеткам, невозможно. Следовательно, план является опорным. В то же время план является невырожденным, т. к. он содержит точно $m + n - 1 = 4 + 5 - 1 = 8$ занятых клеток.

Таблица 7.3 – Построение первоначального опорного плана методом северо-западного угла

Поставщики, A_i	Потребители, B_j					Запасы
	B_1	B_2	B_3	B_4	B_5	a_i
A_1	10 100	7 –	4 –	1 –	4 –	100
A_2	2 100	7 150	10 –	6 –	11	250
A_3	8 –	5 50	3 100	2 50	2	200
A_4	11 –	8 –	12 –	16 50	13 250	300
Потребности b_j	200	200	100	100	250	

Если же занятых клеток будет меньше, чем $m + n - 1$, то следует ввести в любую незанятую клетку объем груза равный нулю и, таким образом, вырожденный план привести к невырожденному.

Найдем общую стоимость составленного плана как сумму произведений объемов перевозок, стоящих в левом углу занятых клеток, на соответствующие стоимости в этих же клетках:

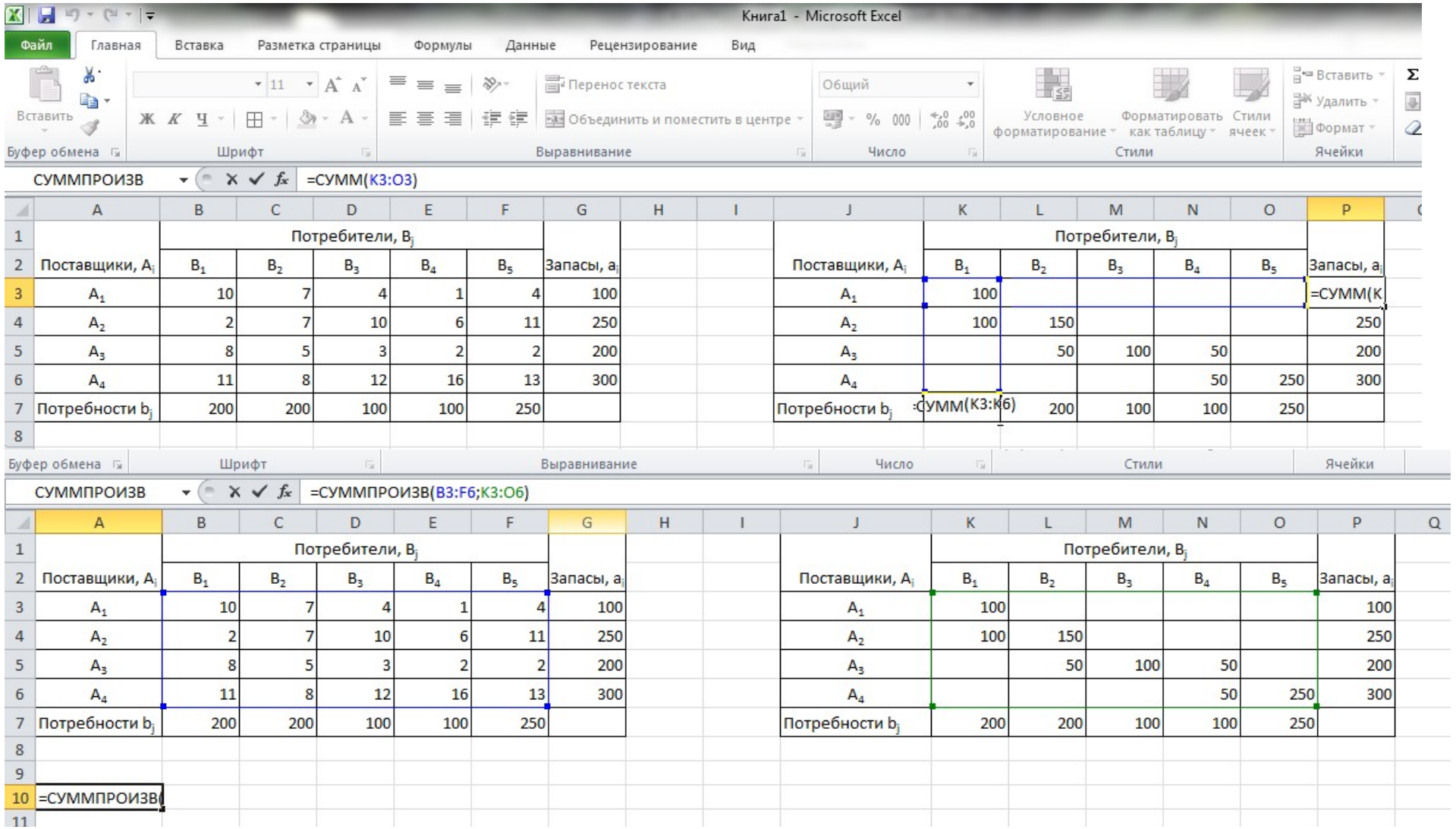
$F = 100 \cdot 10 + 100 \cdot 2 + 150 \cdot 7 + 50 \cdot 5 + 100 \cdot 3 + 50 \cdot 2 + 50 \cdot 16 + 250 \cdot 13 = 6950$(ед. стоимости).

При составлении первоначального опорного плана методом северо-западного угла стоимость перевозки единицы груза не учитывалась, поэтому построенный план далек от оптимального. Построение опорного плана удобно выполнять в табличном процессоре Excel. Для этого

необходимо ввести данные таблицы 7.3 на рабочий лист Excel, затем: эту таблицу скопировать и удалить все числа, оставив название полей; ввести формулы для подсчета сумм перевозок по строкам и столбцам Затем в соответствующие ячейки по строке «Потребности» и столбцу «Запасы»; заполнить ячейки таблицы значениями перевозок.

На рисунке 7.4 представлен рабочий лист Excel с введенными в первой таблице исходными данными задачи: стоимостями, запасами и потребностями, а во второй таблице – формулами для подсчета сумм перевозок по строкам:=СУММ(K3:O3) и столбцам: =СУММ(K3:K6) и найденным опорным планом.

В ячейке A10 введена формула, с помощью которой определяется целевая функция: =СУММПРОИЗВ(B3:F6;K3:O6). Если при составлении опорного плана учитывать стоимость перевозки единицы груза, то план будет значительно ближе к оптимальному.



Поставщики, A_i	Потребители, B_j					Запасы, a_i
	B_1	B_2	B_3	B_4	B_5	
A_1	10	7	4	1	4	100
A_2	2	7	10	6	11	250
A_3	8	5	3	2	2	200
A_4	11	8	12	16	13	300
Потребности b_j	200	200	100	100	250	

Поставщики, A_i	Потребители, B_j					Запасы, a_i
	B_1	B_2	B_3	B_4	B_5	
A_1	100					=СУММ(K
A_2	100	150				250
A_3		50	100	50		200
A_4				50	250	300
Потребности b_j	=СУММ(K3:K6)	200	100	100	250	

Рисунок 7.4 – Построение первоначального опорного плана методом северо-западного угла

Найдем опорный план методом минимальной стоимости. Сущность метода заключается в том, что из всей таблицы стоимостей выбирают наименьшую и в клетку, которая ей соответствует, помещают меньшее из чисел a_i или b_j. Затем из рассмотрения исключают либо строку, соответствующую поставщику, запасы которого полностью израсходованы, либо столбец, соответствующий потребителю, потребности которого полностью удовлетворены, либо и строку, и столбец, если израсходованы запасы поставщика и удовлетворены потребности потребителя. Из оставшейся части таблицы стоимостей снова выбирают наименьшую стоимость, и процесс распределения запасов продолжают, пока все запасы не будут распределены, потребности удовлетворены. Составим с помощью этого метода опорный план уже рассмотренной задачи и запишем ее условие в таблицу 7.5. Выбираем в таблице наименьшую стоимость (это стоимость, помещенная в клетке A_1B_4), так как $a_1=b_4$, 100 ед. груза помещаем в этой клетке и исключаем из рассмотрения первую строку и четвертый столбец. В оставшейся таблице стоимостей наименьшей является стоимость, расположенная в клетке A_2B_1 и в клетке A_3B_5. Заполняем любую из них, например, A_2B_1.

Имеем 200<250, следовательно, записываем в нее 200 и исключаем из рассмотрения столбец B_1. В клетку A_3B_5 записываем 200 ед. и исключаем из рассмотрения строку A_3.

Таблица 7.5 – Построение первоначального опорного плана методом минимальной стоимости

Поставщики, A_i	Потребители, B_j					Запасы a_i
	B_1	B_2	B_3	B_4	B_5	
A_1	10 –	7 –	4 –	1 100	4 –	100
A_2	2 200	7 50	10 –	6 –	11	250
A_3	8 –	5 –	3 –	2 –	2 200	200
A_4	11 –	8 150	12 100–	16 –	13 50	300
Потребности b_i	200	200	100	100	250	

В оставшейся таблице стоимостей снова выбираем наименьшую стоимость и продолжаем процесс до тех пор, пока все запасы не будут распределены, потребности удовлетворены. В результате получен план X=(x_{14}=100; x_{21}=200; x_{22}=50; x_{35}=200; x_{42}=150; x_{43}=100; x_{45}=50), остальные значения переменных равны нулю. План не содержит циклов и состоит из семи положительных перевозок, следовательно, является опорным планом. Определим его стоимость:

$$F = 100 \cdot 1+200 \cdot 2+50 \cdot 7+200 \cdot 2+ 150 \cdot 8+ 100 \cdot 12++50 \cdot 13=4300 \text{ (ед.)}.$$

Стоимость плана перевозок значительно меньше, следовательно, он ближе к оптимальному.

Определим опорный план методом двойного предпочтения. Если таблица стоимостей велика, то перебор всех элементов затруднителен. В этом случае используют метод двойного предпочтения, суть которого заключается в следующем:

В каждом столбце отмечают знаком «V» клетку с наименьшей стоимостью. Затем то же проделывают в каждой строке. В результате некоторые клетки имеют отметку «VV» (рисунок 7.6).

В них находится минимальная стоимость, как по столбцу, так и по строке. В эти клетки помещают максимально возможные объемы перевозок, каждый раз исключая из рассмотрения соответствующие столбцы или строки. Затем распределяют перевозки по клеткам, отмеченным знаком «V». В оставшейся части таблицы перевозки распределяют по наименьшей стоимости.

Таблица 7.6 – Выделенные клетки с минимальной стоимостью

Поставщики, A_i	Потребители, B_j					Запасы a_i
	B_1	B_2	B_3	B_4	B_5	
A_1	10	7	4	VV 1	4	100
A_2	VV 2	7	10	6	11	250
A_3	8	V 5	V 3	V 2	VV 2	200
A_4	11	V 8	12	16	13	300
Потребности b_j	200	200	100	100	250	

Затем сначала заполняем клетки A_1B_4, A_2B_1, A_3B_5, потом клетку A_3B_5. В оставшейся части таблицы последовательно заполняем клетки по минимальной стоимости A_3B_2, A_3B_3, A_3B_4 , A_4B_2, (таблица7.7).

Таблица 7.7 – Опорный план, найденный методом двойного предпочтения

Поставщики, A_i	Потребители, B_j					Запасы a_i
	B_1	B_2	B_3	B_4	B_5	
A_1	10 –	7 –	4 –	VV 1 100	4 –	100
A_2	VV 2 200	7 –	10 –	6 –	11 50	250
A_3	8 –	V 5 –	V 3 –	V 2 –	VV 2 200	200
A_4	11 –	V 8 200	12 100	16 –	13 –	300
Потребности b_j	200	200	100	100	250	

Найдем стоимость полученного опорного плана.

F= 100 ·1+200 ·2 +50 ·10+200 ·2+200 ·8+50 ·12+50 ·13 = 4250 (ед.).

Таким образом, наименьшую стоимость имеет опорный план, полученный методом двойного предпочтения, следовательно, он наиболее близок к оптимальному плану. Однако отсюда не следует вывод, что с помощью метода двойного предпочтения всегда получают лучший первоначальный план по сравнению с методом минимальной стоимости.

Можно показать, что задача линейного программирования двойственная классической транспортной задаче (7.1, 7.2) состоит в максимизации целевой функции:

$$\phi(\alpha_1,\ldots,\alpha_m,\beta_1,\ldots,\beta_n)=\sum_{i=1}^{m=4} a_i\alpha_i+\sum_{j=1}^{n=5} b_j\beta_j \tag{7.5}$$

при ограничениях

$$\alpha_i+\beta_j \le c_{ij}, i=\overline{1,m}; j=\overline{1,n}, \tag{7.6}$$

где числа $\alpha_i, i=\overline{1,m}$ и $\beta_j, j=\overline{1,n}$ называются соответственно потенциалами поставщиков и потребителей. Для того чтобы решение классической транспортной задачи было оптимальным необходимо выполнение следующей теоремы:

Теорема. Если план $X^*=(x_{ij}^*)$ транспортной задачи является оптимальным, то ему соответствует система из $m+n$ чисел $\alpha_i, i=\overline{1,m}$ и $\beta_j, j=\overline{1,n}$, удовлетворяющих условиям:

$$\alpha_i+\beta_j=c_{ij}, \text{ для } x^*_{ij}>0, \tag{7.8}$$

$$\alpha_i+\beta_j \le c_{ij}, \text{ для } x^*_{ij}=0. \tag{7.9}$$

Из теоремы следует: для того, чтобы первоначальный опорный план был оптимальным, необходимо выполнение следующих условий:

- для каждой занятой клетки сумма потенциалов должна быть равна стоимости единицы перевозки, стоящей в этой клетке $\alpha_i+\beta_j=c_{ij}$;
- для каждой незанятой клетки сумма потенциалов должна быть меньше или равна стоимости единицы перевозки, стоящей в этой клетке $\alpha_i+\beta_j \le c_{ij}$.

Если хотя бы одна незанятая клетка не удовлетворяет этому условию, то опорный план не является оптимальным и его можно улучшить, перемещая в эту клетку некоторое количество груза.

Используя первое условие теоремы 7.6, построим систему потенциалов (таблица 7.6), используя таблицу 7.7, предварительно проверив, сколько занятых клеток имеет таблица. Если занятых клеток меньше, чем $m+n-1$ заполняем недостающие клетки нулевыми перевозками.

Систему потенциалов можно построить только для невырожденного плана, который, как известно, содержит $m+n-1$ занятых клеток, поэтому для

него можно составит систему из $m+n-1$ независимых уравнений $\alpha_i + \beta_j = c_{ij}$ с $m+n$ неизвестными.

Уравнений на одно меньше, чем неизвестных, поэтому система является неопределенной и одному неизвестному (обычно α_i) присваивают нулевое значение, после этого остальные потенциалы определяются однозначно.

Пусть известен потенциал α_i, тогда $\beta_j = c_{ij} - \alpha_i$, если известен потенциал β_j, то $\alpha_i = c_{ij} - \beta_j$.

Таким образом, для определения неизвестного потенциала от величины c_{ij} надо вычесть известный потенциал.

Так как в нашем примере $m = 4$, $n = 5$, то для того, чтобы найденный в таблице 7.7 план был невырожденным, он должен содержать 4 + 5 – 1 = 8 занятых клеток.

В таблице 7.7 всего шесть занятых клеток, следовательно, в две не занятые клетки необходимо поместить нулевые перевозки.

Заполним нулевыми перевозками клетки A_2B_2 и A_2B_4, так как во второй строке среди незанятых клеток они имеют минимальную стоимость перевозок, а именно: 6 и 7. Запишем целевую функцию двойственной задачи, которую необходимо максимизировать.

$$\varphi(\alpha_1,...,\alpha_4,\beta_1,...,\beta_5) = \sum_{i=1}^{4} a_i\alpha_i + \sum_{j=1}^{5} b_j\beta_j = 100\cdot\alpha_1 + 250\cdot\alpha_2 + 200\cdot\alpha_3 + 300\cdot\alpha_4 +$$

$$+ 200\cdot\beta_1 + 200\cdot\beta_2 + 100\cdot\beta_3 + 100\cdot\beta_4 + 250\cdot\beta_5 (\max)$$

при ограничениях:

$$\begin{cases} \alpha_1 + \beta_4 \le 1 \\ \alpha_2 + \beta_1 \le 2 \\ \alpha_2 + \beta_2 \le 7 \\ \alpha_2 + \beta_4 \le 6 \\ \alpha_2 + \beta_5 \le 11 \\ \alpha_3 + \beta_5 \le 2 \\ \alpha_4 + \beta_2 \le 8 \\ \alpha_4 + \beta_3 \le 12. \end{cases}$$

Получили систему из восьми неравенств и девяти неизвестных. Выберем в качестве свободной переменной $\alpha_2 = 0$, тогда потенциалы $\beta_1,\beta_2,\beta_4,\beta_5$ определятся однозначно. Запишем найденные значения потенциалов в таблицу 7.8.

Таблица 7.8 – Построение системы потенциалов

Поставщики, A_i	Потребители, B_j					Запасы
	$B_1,\beta_1=2$	$B_2,\beta_2=7$	$B_3,\beta_3=11$	$B_4,\beta_4=6$	$B_5,\beta_5=11$	a_i

$A_1,\alpha_1=-5$	10	7	4	1 100	4	100
$A_2,\alpha_2=0$	2 200	7 0	10	6 0	11 50	250
$A_3,\alpha_3=-9$	8	5	3	2	2 200	200
$A_4,\alpha_4=1$	11	8 200	12 100	16	13	300
Потребности b_i	200	200	100	100	250	

Проверим выполнение условия $\alpha_i + \beta_j \le c_{ij}$ (4.7) теоремы для незанятых клеток, если условие не выполняется, найдем разность $(\alpha_i + \beta_j) - c_{ij}$ и запишем в соответствующую клетку (таблица7.9).

Таблица 7.9 – Проверка является ли найденный план оптимальным

Поставщики, A_i	Потребители, B_j					Запасы
	$B_1,\beta_1=2$	$B_2,\beta_2=7$	$B_3,\beta_3=11$	$B_4,\beta_4=6$	$B_5,\beta_5=11$	a_i
$A_1,\alpha_1=-5$	10	7	4 2	1 100	4 2	100
$A_2,\alpha_2=0$	2 200	7 0	10 1	6 0	11 50	250
$A_3,\alpha_3=-9$	8	5	3	2	2 200	200
$A_4,\alpha_4=1$	11	8 200	12 100	16	13	300
Потребности b_i	200	200	100	100	250	

Условие оптимальности не выполняется для трех клеток A_1B_3, A_1B_5 и A_2B_3. Разности для этих клеток соответственно равны 2, 2, 1. Выберем клетку, в которую необходимо перераспределить перевозку. Для этого определим $\max[(\alpha_i + \beta_j) - c_{ij}] = \max[2,2,1] = 2$, значит любую из клеток A_1B_3, A_1B_5 можно сделать занятой. Возьмем клетку A_1B_5, так как в этом случае легче построить цикл, отметим ее знаком «+». Клетка A_1B_5 присоединяется к занятым клеткам, которых становиться $m+n$. Таким образом, появляется цикл, все вершины которого, за исключением одной, которая находится на клетке, отмеченной знаком «+», лежат на занятых клетках, причем этот цикл единственный.

Находим цикл и, двигаясь от клетки отмеченной знаком «+», поочередно проставляем знаки «–», «+» в клетках с вершинами цикла (таблица 7.10). Находим $\min x_{ij} = \min[100,50] = 50$ – минимальное значение среди перевозок, стоящих в вершинах цикла, отмеченных знаком «–».

Таблица 7.10 – Построение цикла

Поставщики, A_i	Потребители, B_j					Запасы
	$B_1,\beta_1=2$	$B_2,\beta_2=7$	$B_3,\beta_3=11$	$B_4,\beta_4=6$	$B_5,\beta_5=11$	a_i
$A_1,\alpha_1=-5$	10	7	4	1 – 100	4 +	100
$A_2,\alpha_2=0$	2	7	10	6 +	11 –	250

	200	0		0	50	
A_3,α_3=-9	8	5	3	2	2 200	200
A_4,α_4=1	11	8 200	12 100	16	13	300
Потребности b_i	200	200	100	100	250	

К перевозкам, стоящим в занятых клетках, прибавляем 50, а из перевозок, стоящих в незанятых клетках, вычитаем 50, получаем новый план (таблица 7.11).

Таблица 7.11 – Новый опорный план

Поставщики, A_i	Потребители, B_j					Запасы a_i
	B_1,β_1=2	B_2,β_2=7	B_3,β_3=11	B_4,β_4=6	B_5,β_5=9	
A_1,α_1=-5	10	7	4 2	1 50	4 50	100
A_2,α_2=0	2 200	7 0	10 1	6 50	11	250
A_3,α_3=-7	8	5	3	2	2 200	200
A_4,α_4=1	11	8 200	12 100	16	13	300
Потребности b_i	200	200	100	100	250	

Пересчитываем потенциалы, так клетка A_2B_5 стала не занятой, а клетка A_1B_5 прибавилась к занятым клеткам. Проверяем является ли найденный план оптимальным. Условие оптимальности не выполняется для двух клеток A_1B_3 и A_2B_3. Разности для этих клеток соответственно равны 2, 1. Выберем клетку, в которую необходимо перераспределить перевозку. Для этого определим $\max[(\alpha_i + \beta_j) - c_{ij}] = \max[2,1] = 2$, значит клетку A_1B_3 надо сделать занятой. Отметим ее знаком «+». Клетка A_1B_3 присоединяется к занятым клеткам, которых становиться $m+n$. Таким образом, появляется цикл все вершины которого, за исключением одной, которая находится на клетке, отмеченной знаком «+», лежат на занятых клетках, причем этот цикл единственный (таблица 7.12). Находим цикл и, двигаясь от клетки отмеченной знаком «+», поочередно проставляем знаки «–», «+» в клетках с вершинами цикла.

Таблица 7.12– Построение цикла

Поставщики, A_i	Потребители, B_j					Запасы a_i
	B_1,β_1=2	B_2,β_2=7	B_3,β_3=11	B_4,β_4=6	B_5,β_5=9	
A_1,α_1=-5	10	7	4 +	1 – 50	4 50	100
A_2,α_2=0	2 200	7– 0	10 1	6 + 50	11	250
A_3,α_3=-7	8	5	3	2	2 200	200
A_4,α_4=1	11	8+	12–	16	13	300

		200	100			
Потребности b_i	200	200	100	100	250	

Определяем $\min x_{ij} = \min[0,100,50] = 0$ – минимальное значение среди перевозок, стоящих в вершинах цикла, отмеченных знаком «–». Значит, нулевую перевозку необходимо переместить в клетку A_1B_3, сделав ее занятой, и опять пересчитать потенциалы (таблица 7.11).

Проверяем найденный план на оптимальность. Для всех незанятых клеток условие оптимальности выполняется. Значит, найденный план является оптимальным.

Находим значение целевой функции.

Таблица 7.13 – Построение новой системы потенциалов

Поставщики, A_i	Потребители, B_j					Запасы a_i
	$B_1,\beta_1=2$	$B_2,\beta_2=5$	$B_3,\beta_3=9$	$B_4,\beta_4=6$	$B_5,\beta_5=9$	
$A_1,\alpha_1=-5$	10	7	4 0	1 50	4 50	100
$A_2,\alpha_2=0$	2 200	7	10	6 50	11	250
$A_3,\alpha_3=-7$	8	5	3	2	2 200	200
$A_4,\alpha_4=3$	11	8 200	12 100	16	13	300
Потребности b_i	200	200	100	100	250	

$F_{min} = 1\cdot50+4\cdot50+6\cdot50+2\cdot200+2\cdot200+8\cdot200+12\cdot100=4150.$

$\Psi_{max}=-5\cdot100+0\cdot250-7\cdot200+3\cdot300+2\cdot200+5\cdot200+9\cdot100+6\cdot100+9\cdot250 =4150.$

Так как $F_{min}=\Psi_{max} = 4150$, т.е. целевая функция и прямой, и двойственной задачи имеет одно и тоже значение и выполняется условие оптимальности для незанятых клеток таблицы 4.13, значит, найденный план является оптимальным.

Ответ: $F_{min}= 4150$ при: $x_{14} = 50$, $x_{15} =50$, $x_{12} =200$, $x_{14} =50$, $x_{35} =200$, $x_{42} =200$, $x_{43} =100$.

Проверим полученный результат, решив задачу с помощью инструмента «Поиск решения» табличного процессора Excel.

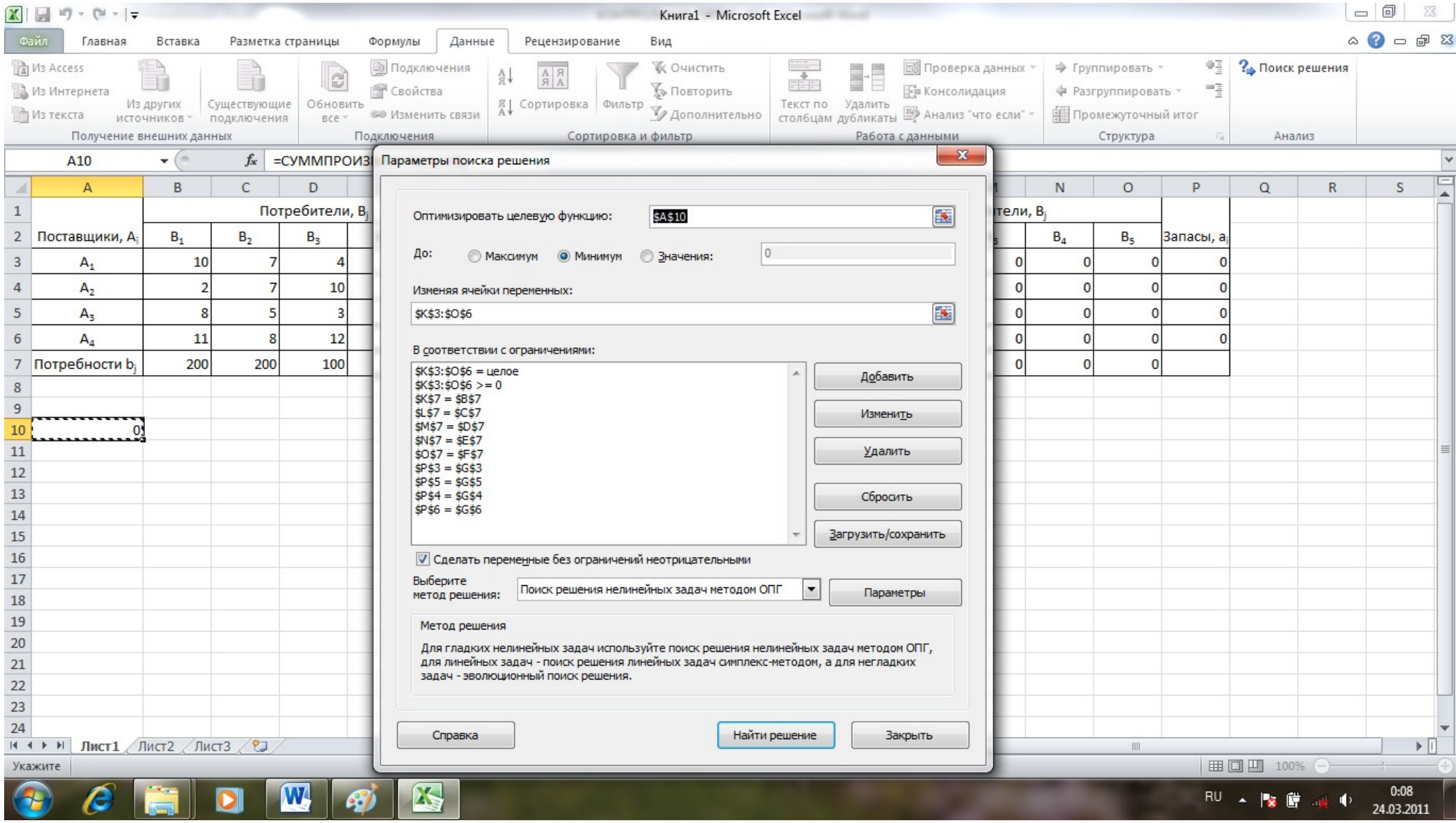


Рисунок 7.5 – Рабочий лист с введенными данными и окно диалога «Поиск решений»

Рабочий лист с решением задачи, представленной матрицей планирования (таблица 7.2), в системе компьютерной математики Mathcad представлен на рисунке 7.6.

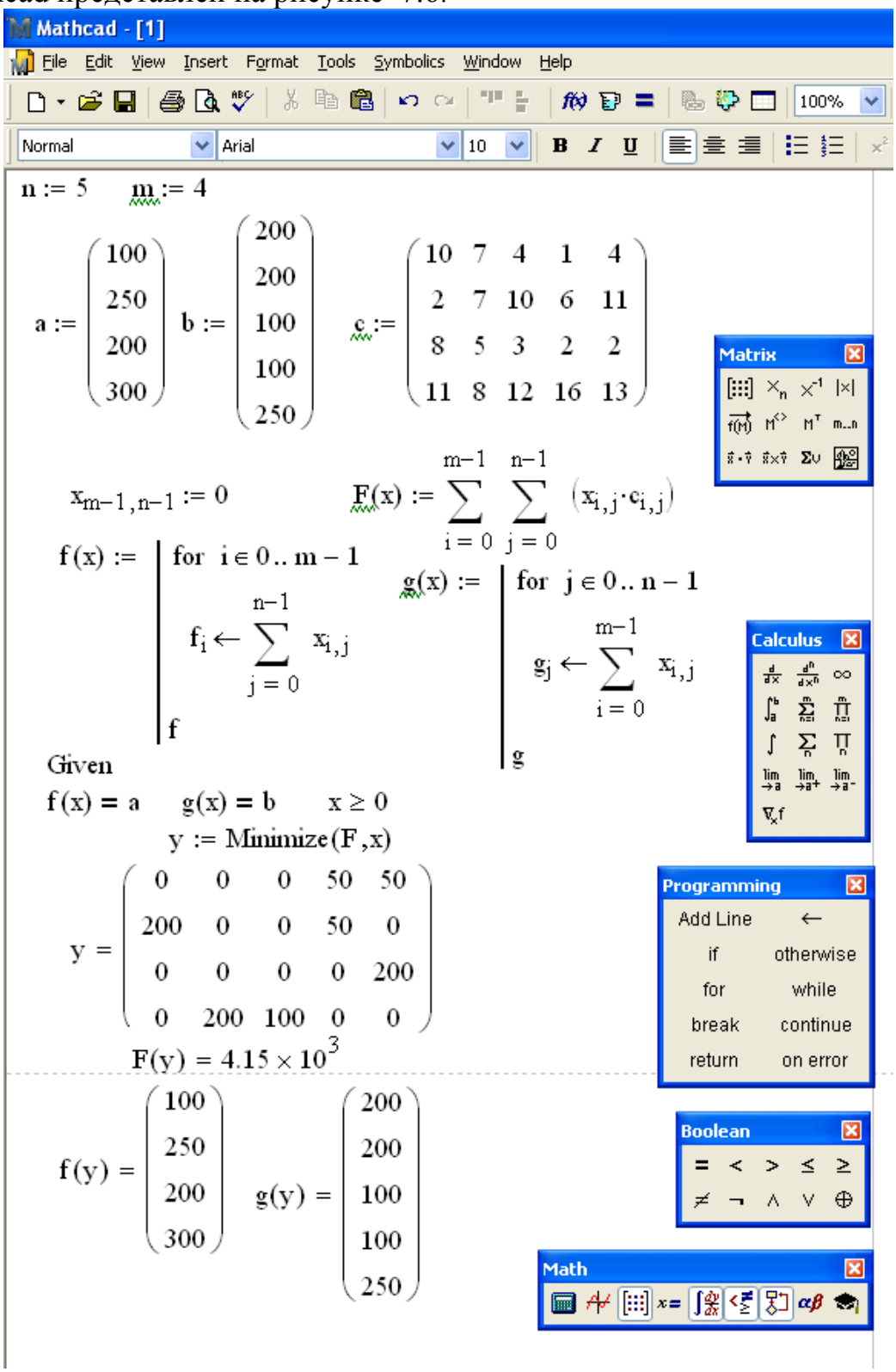


Рисунок 7.6 – Алгоритм решения задачи в Mathcad

Аппаратура и материалы. Для выполнения лабораторной работы необходимо использовать следующее: аппаратное обеспечение: персональный компьютер и программное обеспечение: операционную

систему Windows 10 и выше; Microsoft Office13 и выше, системы компьютерной математики Machcad и MATLAB R2017a и выше.

Указания по технике безопасности. Самостоятельно не производить установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

Выполните предложенные задания.

Задание 7.1. Информационные модели принятия решений в условиях определенности.

Найти первоначальный опорный план (методами: северо-западного угла; минимальной стоимости; двойного предпочтения) следующей задачи**:** Пусть имеется три пункта отправления (поставщика) A1, A2, A3 и четыре пункта назначения (потребителя) B1, B2, B3, B4. Известны расстояния (в км) от каждого поставщика до каждого потребителя. Эти расстояния приведены ниже в таблице 7.14 в левом верхнем углу клеток. Требуется составить такой план перевозок груза, при котором общий грузооборот (в т.км) был бы минимальным.

Таблица 7.14 – Матрица стоимостей

Поставщики	Потребители				Запасы
	B_1	B_2	B_3	B_4	a_i
A_1	6	12	20	22	1500
A_2	10	18	18	17	2800
A_3	1	4	6	11	1700
Потребности *bj*	1830	1160	1610	1400	

1. Постройте математическую модель транспортной задачи.
2. Найдите опорный план методами: северо-западного угла; минимальной стоимости; двойного предпочтения.
3. Постройте систему потенциалов и проверьте найденные планы на оптимальность.
4. Найдите оптимальный план:
   - с помощью инструмента «Поиск решения» табличного процессора Excel;
   - с помощью системы компьютерной математики Mathcad;
   - методом потенциалов, взяв за первоначальный любой из опорных планов, не являющихся оптимальным.
5. Сделайте вывод.

Задание 6. 2. Индивидуальное задание: Используя таблицу из своего варианта (таблица 7.15), постройте математическую модель заданной экономической системы. Постройте опорный план методами: северо-западного угла; минимальной стоимости; двойного предпочтения,

найдите оптимальный план с помощью инструмента «Поиск инструмента».

Таблица 7.15 – Варианты первого задания

Вариант №	Условие задачи					
1.	Поставщики	Потребители				Запасы a_i
		B_1	B_2	B_3	B_4	
	A_1	5	2	8	3	120
	A_2	7	3	9	4	180
	A_3	3	2	1	2	160
	Потребности b_j	40	160	140	120	
2.	Поставщики	Потребители				Запасы a_i
		B_1	B_2	B_3	B_4	
	A_1	3	2	1	3	210
	A_2	8	7	9	10	190
	A_3	4	2	10	12	100
	Потребности b_j	130	100	150	120	
3.	Поставщики	Потребители				Запасы a_i
		B_1	B_2	B_3	B_4	
	A_1	5	2	8	3	120
	A_2	7	3	9	4	180
	A_3	3	2	1	2	150
	Потребности b_j	130	100	170	50	
4.	Поставщики	Потребители				Запасы a_i
		B_1	B_2	B_3	B_4	
	A_1	2	1	4	3	120
	A_2	6	4	3	7	170
	A_3	4	2	5	9	110
	Потребности b_j	50	00	150	110	
5.	Поставщики	Потребители				Запасы a_i
		B_1	B_2	B_3	B_4	
	A_1	5	2	8	3	110
	A_2	7	3	9	4	180
	A_3	3	2	1	2	210
	Потребности b_j	140	100	140	120	
6.	Поставщики	Потребители				Запасы a_i
		B_1	B_2	B_3	B_4	
	A_1	5	2	8	3	100
	A_2	7	3	9	4	160
	A_3	3	2	1	2	140
	Потребности b_j	100	200	50	50	
7.	Поставщики	Потребители				Запасы a_i
		B_1	B_2	B_3	B_4	
	A_1	3	2	1	4	110
	A_2	9	6	4	5	150
	A_3	6	2	2	7	140

	Потребности b_j	20	140	130	110	
8.	Поставщики	Потребители				Запасы
		B_1	B_2	B_3	B_4	a_i
	A_1	4	2	4	3	130
	A_2	7	4	9	4	170
	A_3	11	12	3	10	120
	Потребности b_j	70	140	110	100	
9.	Поставщики	Потребители				Запасы
		B_1	B_2	B_3	B_4	a_i
	A_1	1	5	6	3	150
	A_2	7	4	3	8	100
	A_3	8	2	3	1	150
	Потребности b_j	180	60	40	120	
10.	Поставщики	Потребители				Запасы
		B	B_2	B_3	B_4	a_i
	A_1	1	5	6	3	120
	A_2	5	9	3	4	180
	A_3	3	2	7	2	100
	Потребности b_j	80	60	140	120	
11.	Поставщики	Потребители				Запасы
		B_1	B_2	B_3	B_4	a_i
	A_1	2	3	2	7	100
	A_2	1	3	9	4	250
	A_3	2	6	3	1	100
	Потребности b_j	80	160	140	120	
12.	Поставщики	Потребители				Запасы
		B_1	B_2	B_3	B_4	a_i
	A_1	2	7	5	1	100
	A_2	4	2	3		190
	A_3	9	6	2	8	150
	Потребности b_j	80	120	100	140	
13.	Поставщики	Потребители				Запасы
		B_1	B_2	B_3	B_4	a_i
	A_1	1	4	8	6	140
	A_2	8	2	4	5	150
	A_3	5	1	3	7	150
	Потребности b_j	80	120	140	100	
14.	Поставщики	Потребители				Запасы
		B_1	B_2	B_3	B_4	a_i
	A_1	2	7		7	140
	A_2	7	1	8	3	170
	A_3	5	2	5	1	150
	Потребности b_j	80	100	140	140	
15.	Поставщики	Потребители				Запасы
		B_1	B_2	B_3	B_4	a_i
	A_1	3	8	2	7	140
	A_2	7	2	7	1	190
	A_3	4	3	5	4	110
	Потребности b_j	40	120		140	

				40		
16.	Поставщики	Потребители				Запасы
		B_1	B_2	B_3	B_4	a_i
	A_1	3	7	2	7	140
	A_2	7	3	1	3	190
	A_3	5	2	5	4	150
	Потребности b_j	80	120	140	140	
17.	Поставщики	Потребители				Запасы
		B_1	B_2	B_3	B_4	a_i
	A_1	3	7	2	7	140
	A_2	1	8	3	9	150
	A_3	5	1	2	1	150
	Потребности b_j	80	100	120	140	
18.	Поставщики	Потребители				Запасы
		B_1	B_2	B_3	B_4	a_i
	A_1	1	6	3	3	140
	A_2	5	2	4	1	240
	A_3	4	8	6	5	150
	Потребности b_j	130	120	140	140	
19.	Поставщики	Потребители				Запасы
		B_1	B_2	B_3	B_4	a_i
	A_1	2	1	3	5	100
	A_2	7	3	1	3	190
	A_3	5	7	6	4	190
	Потребности b_j	100	120	120	140	
20.	Поставщики	Потребители				Запасы
		B_1	B_2	B_3	B_4	a_i
	A_1	3	8	2	7	100
	A_2	1	10	3	9	170
	A_3	5	11	2	1	170
	Потребности b_j	120	100	120	100	

Содержание отчета и его форма

Подготовьте отчет, в котором опишите технологию решения задач, используя задания своего варианта. Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) условия выполненных заданий и их решение;
4) выводы;
5) ответы на контрольные вопросы.

Вопросы для защиты работы:

1. Сформулировать транспортную задачу в общем виде.
2. Сформулировать алгоритм нахождения первоначального опорного плана транспортной задачи методом северо-западного угла.
3. Сформулировать алгоритм нахождения первоначального опорного плана транспортной задачи методом минимальной стоимости.

4. Сформулировать алгоритм нахождения первоначального опорного плана транспортной задачи методом двойного предпочтения.

5. Как проверить найденный план на оптимальность?

ЛАБОРАТОРНАЯ РАБОТА 8
Элементы теории игр в задачах моделирования, поиск оптимальной смешанной стратегии

Цель и содержание: Научиться строить и анализировать модели задач принятия решения в условиях неопределенности.

Организационная форма занятий: решение проблемных задач, разбор конкретных ситуаций

Вопросы для обсуждения на лабораторном занятии: построение, решение и анализ моделей задач теории игр.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Изучите теоретический материал по данной теме, используя материал, изложенный ниже.

Теория игр может применяться в различных сферах искусственного интеллекта:

- Мультиагентные системы искусственного интеллекта.
- Имитация и Усиление обучения.
- Обучение противника в Генеративных Состязательных Сетях (GAN).

Теория игр может также использоваться для описания многих ситуаций в нашей повседневной жизни и моделях машинного обучения

Игры занимают ключевое место в эволюции искусственного интеллекта (ИИ), так как в настоящее время игровые среды становятся популярным тренировочным механизмом в таких областях, как обучение с подкреплением (reinforcement learning) и имитационное обучение (imitation learning). В контексте искусственного интеллекта и систем глубокого обучения теория игр обеспечивает некоторые ключевые возможности, необходимые в многоагентных средах, в которых различные программы ИИ должны взаимодействовать или соревноваться друг с другом для достижения какой-либо цели.

В теории, любая многоагентная система ИИ может работать с геймифицированными взаимодействиями участников. Раздел математики, который формулирует игровые принципы, называется теорией игр. [https://medium.datadriveninvestor.com/game-theory-for-data-scientists-e44048a7e935]

Теорией игр называют математическую дисциплину, занимающуюся обоснованием оптимальных решений, принимаемых в тех или иных конфликтных ситуациях. Под ***конфликтной ситуацией*** понимается такая ситуация, в которой необходимо принимать решения в условиях, когда целям одной стороны противостоят противоположные цели другой стороны. Подобные ситуации характерны для военных

действий, игровых видов спорта, практической деятельности. Во всех подобных случаях происходит столкновение противоположных интересов, принятие решения каждой из сторон связано с преодолением конфликта. Принятие решения в конфликтной ситуации затрудняется из-за неопределенности поведения противника. Теория игр позволяет точным математическим языком описать правила игры, дать формулы, оценивающие ее ситуации и указывающие правильный путь к победе. Эта теория позволяет с некоторой долей точности учесть многие обстоятельства, влияющие на борьбу и предсказать ее исход с определенной степенью достоверности.

Основные понятия теории игр. Классификация и описание игр

В основе теории игр лежит ***конфликт***, который выступает как нечто формализуемое, объективно анализируемое и просчитываемое при необходимости достижения цели. Для математического анализа конфликта строится упрощенная модель конфликтной ситуации, которую называют ***игрой***. Ее участников по традиции называют ***игроками.*** Игру ведут по определенным правилам. Результат игры называют ***выигрышем*** или ***платежом***. Одну реализацию игры – ***партией.*** Выбор игроком того или иного действия – ***ходом.*** Различают два вида ходов – личные и случайные. ***Личный ход*** предполагает сознательный выбор игроком того или иного действия, разрешенного правилами игры. ***Случайный ход*** не зависит от воли игрока и является результатом того или иного случайного процесса (например, сдача карт). Игры, состоящие только из случайных ходов, называются ***азартными***. Игры, в которых имеются личные ходы, называются ***стратегическими***. Существуют также стратегические игры, состоящие, как из личных, так и случайных ходов. Теория игр занимается стратегическими играми. Каждый игрок располагает конечным или бесконечным набором допустимых решений, называемых ***стратегиями***. Выигрыш каждого игрока определяется его платежной функцией, значения которой зависят от стратегий всех участников игры.

Фактически игра представляет собой совокупность правил, известных всем игрокам. Эти правила, с одной стороны, определяют множества стратегий игроков, а с другой – последствия и выигрыши в результате выбора каждой из стратегий. Задача теории игр определить такую стратегию игрока, при которой его шансы на выигрыш оказались бы наибольшими.

В теории игр рассматриваются игры двух и игры многих лиц, игры, в которых каждый игрок может делать лишь один ход, и игры с множеством ходов. Стратегии игроков могут использовать или не использовать случайный механизм. В конфликтных ситуациях, в которых допустимо применение случайного механизма, последствия игры определяются средним выигрышем при многократном повторении игры. В таких играх выбор стратегии предусматривает также выбор функции распределения используемого случайного механизма.

Классификацию игр проводят по различным признакам:

- по числу игроков;
- по числу стратегий;
- по свойствам платежной функции;
- по характеру предварительной договоренности между игроками.

Игру, в которой участвует *n* игроков, называют ***игрой с n участниками.*** В игре могут возникать как конфликтные ситуации, так и необходимость в координированных действиях ***(кооперация).*** Если в игре участвует не менее трех игроков, то могут создаваться ***коалиции,*** т.е. группы из двух или более игроков, имеющих общую цель и координирующих свои стратегии.

По количеству стратегий различают ***игры конечные*** **и** ***бесконечные.*** Если хотя бы один из игроков располагает бесконечным множеством стратегий, то игру называют бесконечной. Если же каждый из игроков располагает конечным множеством стратегий, то игру называют конечной.

Еще один способ классификации игр – по свойствам платежной функции. В ***игре с нулевой суммой*** общая сумма выигрышей всех игроков равна нулю. В игре с нулевой суммой и двумя участниками выигрыш одного из них равен проигрышу другого. Таким образом, в играх с нулевой суммой существует конфликт между игроками и поэтому их называют также ***антагонистическими играми.*** В общем случае в игре с нулевой суммой, как правило, имеют место и конфликты, и согласованные действия игроков. Прямой противоположностью играм с нулевой суммой являются ***игры*** двух игроков ***с постоянной разностью,*** в которых оба игрока выигрывают или проигрывают одновременно. Поэтому игрокам выгодно действовать согласованно.

В зависимости от характера предварительной договоренности между игроками различают ***кооперативные*** **и** ***некооперативные игры.*** Игра кооперативная, если до ее начала игроки образуют коалиции и принимают взаимообязывающие соглашения о координации своих стратегий. В противном случае игра будет некооперативной.

Существуют два основных способа описания и анализа любой конкретной игры.

Первый способ предполагает:

- перечисление ходов, которые могут делать игроки;
- определение информации, которой располагают игроки в процессе игры;
- определение возможных вариантов действий игроков;
- указание предельных размеров платежей в конце игры.

Игру, описанную подобным образом, называют ***игрой в развернутой*** или ***экстенсивной форме,*** а само описание, как правило, составляют в виде ***дерева игры***. Игры в развернутой форме называют

также ***позиционными играми.***

Игру в развернутой форме называют ***игрой с полной информацией***, если в ней нельзя делать одновременно несколько ходов и если участникам известны выборы, сделанные при предшествующих ходах, включая и случайные ходы. Примером игры с полной информацией являются шахматы. Покер представляет собой ***игру с неполной информацией,*** так как игрокам неизвестно, какие карты находятся на руках у противника.

Второй способ описания игры предполагает рассмотрение всех возможных стратегий каждого игрока и определение платежей, соответствующих любым возможным комбинациям стратегий всех игроков. Игру, описанную вторым способом, обычно называют игрой **в *нормальной форме.*** Естественно, зная развернутую форму игры, всегда можно представить ее в нормальной форме.

Нормальная форма игры двух участников состоит из двух матриц**,** содержащих суммы выигрышей и проигрышей каждого из игроков для любой из возможных пар стратегий. Обычно эти две матрицы объединяются в одну, называемую ***платежной матрицей игры.***

Платежная матрица игры. Нижняя и верхняя цена игры. Принцип минимакса (максимина)

Рассмотрим платежную матрицу игры на примере простейшей игры. Простейшие игры – это игры двух лиц с нулевой суммой, т.е. ***парные игры***, в которых выигрыш одного игрока совпадает с проигрышем другого.

Результаты конечной игры (парной) с нулевой суммой можно задавать матрицей, строки и столбцы которой соответствуют различным отражениям, а ее элементы – соответствующие выигрыши одной стороны (равные проигрышам другой стороны). Эта матрица называется платежной матрицей или матрицей игры. При этом удобно проигрыш первой стороны рассматривать как ее отрицательный выигрыш, а выигрыш второй – как ее отрицательный проигрыш. Если первая сторона имеет m стратегий, а вторая n, то говорят, что мы имеем дело с игрой m x n. Итак, пусть первый игрок имеет m стратегий $i = 1,2,..., m$, второй имеет n стратегий $j = 1,2,..., n$. Каждой паре стратегий (i,j) поставлено в соответствие число a_{ij}, выражающее выигрыш первого игрока за счёт второго игрока, если первый игрок примет свою i-ю стратегию, а второй – свою j-ю стратегию. В случае выигрыша первого игрока – $a_{ij} > 0$, а в случае его проигрыша – $a_{ij} < 0$.

Каждая стратегия игрока $i = \overline{1,m}$; $j = \overline{1,n}$ часто называется чистой стратегией.

Если рассмотреть матрицу:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ \cdots\cdots & \cdots\cdots & \cdots & \cdots\cdots & \cdots & \cdots\cdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \cdots\cdots & \cdots\cdots & \cdots & \cdots\cdots & \cdots & \cdots\cdots \\ a_{m1} & a_{m2} & \cdots & a_{mj} & \cdots & a_{mn} \end{pmatrix},$$

то проведение каждой партии матричной игры с матрицей A сводится к выбору первым игроком *i*-й строки, а вторым игроком *j*-го столбца и получения первым игроком (за счёт второго игрока) выигрыша a_{ij}.

Главным в исследовании игр является понятие оптимальных стратегий игроков. В это понятие интуитивно вкладывается такой смысл: стратегия игрока является оптимальной, если применение этой стратегии обеспечивает ему наибольший гарантированный выигрыш при всевозможных стратегиях другого игрока. Исходя из этих позиций, первый игрок исследует матрицу выигрышей A следующим образом: для каждого значения i ($i = \overline{1,m}$) определяется минимальное значение выигрыша в зависимости от применяемых стратегий второго игрока $\min\limits_j a_{ij}$, ($i = \overline{1,m}$), т.е. определяется минимальный выигрыш для первого игрока при условии, что он примет свою *i*-ю чистую стратегию, затем из этих минимальных выигрышей отыскивается такая стратегия, при которой этот минимальный выигрыш будет максимальным, т.е. находится максимин: $\max\limits_i \min\limits_j a_{ij} = \underline{\alpha}$ – нижняя цена игры.

Число $\underline{\alpha}$, называется ***нижней ценой игры*** и показывает, какой минимальный выигрыш может гарантировать себе первый игрок, применяя свои чистые стратегии при всевозможных действиях второго игрока.

Второй игрок при оптимальном своём поведении должен стремиться по возможности за счёт своих стратегий максимально уменьшить выигрыш первого игрока. Поэтому для второго игрока отыскивается $\max\limits_i a_{ij}$, т.е. определяется max выигрыш первого игрока при условии, что второй игрок применит свою *j*-ю чистую стратегию, затем второй игрок отыскивает такую свою стратегию, при которой первый игрок получит min выигрыш, т.е. находит минимакс: $\min\limits_j \max\limits_i a_{ij} = \overline{\alpha}$.

Число $\overline{\alpha}$ называется ***верхней ценой игры*** и показывает, какой максимальный выигрыш за счёт своих стратегий может себе гарантировать первый игрок.

Другими словами, применяя свои чистые стратегии, первый игрок может обеспечить себе выигрыш не меньше $\underline{\alpha}$, а второй игрок за счёт

применения своих чистых стратегий может не допустить выигрыш первого игрока больше, чем $\bar{\alpha}$.

Стратегия, соответствующая минимаксу, называется минимаксной стратегией.

Принцип, диктующий игрокам выбор наиболее «осторожных» минимаксной и максиминной стратегий, называется принципом минимакса.

Если в игре с матрицей А $\underline{\alpha} = \bar{\alpha}$, то говорят, что эта игра имеет седловую точку**,** т.е**.** решение в чистых стратегиях и цену игры $\upsilon = \underline{\alpha} = \bar{\alpha}$.

Седловая точка – это пара чистых стратегий соответственно первого и второго игроков, при которых достигается равенство $\underline{\alpha} = \bar{\alpha}$. В это понятие вложен следующий смысл: если один из игроков придерживается стратегии, соответствующей седловой точке, то другой игрок не сможет поступить лучше, чем придерживаться стратегии, соответствующей седловой точке.

Таким образом, седловой элемент является минимальным в i-й строке и максимальным в j-м столбце в матрице А. Отыскание седловой точки матрицы А происходит следующим образом: в матрице А последовательно в каждой строке находят минимальный элемент и проверяют, является ли этот элемент максимальным в своём столбце. Если да, то он и есть седловой элемент, а пара стратегий, ему соответствующая, образует седловую точку. Пара чистых стратегий первого и второго игроков, образующая седловую точку и седловой элемент a_{ij} , называется ***решением игры***.

Пример 1

$$\begin{matrix} & & & & \min\limits_j a_{ij} \\ & & & & \| \\ \mathrm{A}= & \begin{pmatrix} 5 & 1 & 2 \\ 4 & 9 & 8 \\ 6 & 9 & 6 \end{pmatrix} & & \begin{matrix} 1 \\ 4 \\ 6 \end{matrix} \end{matrix} \Big\} \max_i \min_j a_{ij} = 6$$

$$\max_i a_{ij} = \underbrace{6 \quad 9 \quad 8}_{\min\limits_j \max\limits_i a_{ij} = 6.}$$

Седловой точкой является пара ($i = 3; j = 1$), при которой $\upsilon = \underline{\alpha} = \bar{\alpha} = 6$.

Заметим, что, хотя выигрыш в ситуации (3;3) также равен 6, однако она не является седловой точкой, т.к. этот выигрыш не является максимальным среди выигрышей третьего столбца.

Пример 2

$$H=\begin{matrix}\min\limits_{j} a_{ij}\\ \begin{pmatrix}1 & 3\\ 4 & 2\end{pmatrix}\\ \max\limits_{i} a_{ij}\end{matrix}\begin{matrix}\\ \begin{matrix}\rightarrow & 1\\ \rightarrow & 2\end{matrix}\\ \\ \underbrace{4}\quad\underbrace{3}\end{matrix}\begin{matrix}\\ \\ \max\limits_{i}\min\limits_{j} a_{ij}=2\\ \\ \min\limits_{j}\max\limits_{i} a_{ij}=3.\end{matrix}$$

Из анализа матрицы выигрышей видно, что $\underline{\alpha}<\overline{\alpha}$, т.е. данная матрица не имеет седловой точки.

Итак, исследование в матричных играх начинается с нахождения её седловой точки в чистых стратегиях. Если матричная игра имеет седловую точку в чистых стратегиях, то нахождением этой седловой точки заканчивается исследование игры. Если же в игре нет седловой точки в чистых стратегиях, то можно найти нижнюю и верхнюю чистые цены этой игры, которые указывают, что первый игрок не должен надеяться на выигрыш больший, чем верхняя цена игры, и может быть уверен в получении выигрыша не меньше нижней цены игры. Улучшение решений матричных игр следует искать в использовании секретности применения чистых стратегий и возможности многократного повторения игр в виде партии. Этот результат достигается путём применения чистых стратегий случайно, с определённой вероятностью.

В игре двух лиц с нулевой суммой и конечным числом стратегий у каждого игрока под оптимальными стратегиями естественно понимать стратегии, при которых достигается максимально возможный гарантированный выигрыш в наименее благоприятных условиях. Этот принцип называется принципом максимина. В игре двух лиц с нулевой суммой существование оптимальной (максимальной) стратегии в каждой отдельной партии гарантируются только в играх с полной информацией. В игре с неполной информацией максимальная стратегия обеспечивается лишь при применении случайного механизма для выбора очередного хода и многократном повторении игры. В этом случае говорят об использовании смешанных стратегий.

В более сложных играх поиск оптимальных стратегий поведения связан с понятием рационального, справедливого решения игры (так называемый принцип в оптимальной игре), определяемого в зависимости от конкретных содержательных особенностей конфликта.

Принципы оптимальности в теории игр вводятся как некоторые правдоподобные описания представления о фактическом поведении игроков, принимающих решения в условиях конфликта или неопределенности. Все предложенные до сих пор принципы оптимальности прямо или косвенно отражают идею устойчивости ситуации, удовлетворяющей этим принципам. Однако в различных

принципах оптимальности интуитивно оправданная идея устойчивости реализуется по-разному.

Многие принципы оптимального поведения игроков, представляющихся на первый взгляд вполне разумными, в действительности могут оказаться нереализуемыми и поэтому неприменимыми. В связи с этим изучение условий, при которых те или иные интуитивно оправданные принципы оптимальности оказываются реализуемыми, и доказательства соответствующих теорем существования решения игры являются центральной проблемой теории игр, тесно связанной с проблемой выбора из множества допустимых альтернатив и проблемой группового выбора.

Поиск оптимальной смешанной стратегии. Связь теории с линейным программированием

Так как каждый раз применение игроком одной чистой стратегии исключает применение другой, то чистые стратегии являются несовместными событиями. Кроме того, они являются единственными возможными событиями.

Смешанной стратегией игрока называется полный набор вероятностей применения его чистых стратегий.

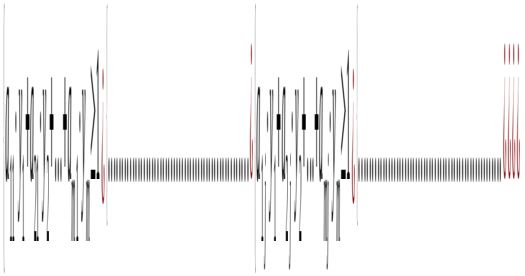
Рассмотрим поиск оптимальной смешанной стратегии на примере матрицы, приведенной выше. Пусть первый игрок имеет m чистых стратегий. Для получения наибольшего эффекта он должен использовать все или некоторые из этих стратегий случайным образом, но не с одинаковыми, а с разными (специально вычисленными) вероятностями. Пусть первая стратегия используется с вероятностью p_1, вторая с вероятностью p_2 и т.д. Говорят, что первый игрок применил свою смешанную стратегию $S_1(p_1,p_2,...,p_m)$. Аналогично, смешанная стратегия второго игрока, который имеет n чистых стратегий – $S_2(q_1,q_2,...,q_n)$.

Для определения оптимальной смешанной стратегии первого игрока необходимо найти вероятности: $p_1,p_2,...,p_m$ применения соответствующих чистых стратегий и цену игры ν, которая больше нижней цены игры, но меньше верхней цены игры. При этом $\sum_{i=1}^{m} p_i = 1$.

Чистая стратегия есть частный случай смешанной стратегии. Действительно, если в смешанной стратегии какая-либо i-я чистая стратегия применяется с вероятностью 1, то все остальные чистые стратегии не применяются. И эта i-я чистая стратегия является частным случаем смешанной стратегии. Для соблюдения секретности каждый игрок применяет свои стратегии независимо от выбора другого игрока.

Допустим, второй игрок выбирает свою первую чистую стратегию, тогда средний выигрыш первого игрока будет равен: $a_{11} \cdot p_1 + a_{21} \cdot p_2 + \ldots + a_{i1} \cdot p_i + \ldots + a_{m1} \cdot p_m \geq \nu$, используем первый столбец платежной матрицы. Если второй игрок выбирает вторую чистую

стратегию, то средний выигрыш первого игрока составит: $a_{12}\cdot p_1+a_{22}\cdot p_2+\ldots+a_{i2}\cdot p_i+\ldots+a_{m2}\cdot p_m\geq v$, используем второй столбец платежной матрицы и т.д., всего получится система, состоящая из n неравенств. Разделим каждое неравенство системы и уравнение $\sum_{i=1}^{m} p_i = 1$ на ν. Введем обозначения: $y_1=\frac{p_1}{v}, y_2=\frac{p_2}{v},\ldots,y_m=\frac{p_m}{v}$, в результате получим:



и $y_1+y_2+\ldots+y_m=\frac{1}{v}$. Для того, чтобы цена игры ν была как можно больше необходимо, чтобы 1/ν была как можно меньше.

Поскольку первый игрок стремится найти такие значения y_i и, следовательно, p_i , чтобы цена игры υ была максимальной, то решение первой задачи сводится к нахождению таких неотрицательных значений y_i $\left(i=\overline{1,m}\right)$, при которых $\sum_{i=1}^{m} y_i \to \min$, $\sum_{i=1}^{m} a_{ij}\, y_i \geq 1$.

Поскольку второй игрок стремится найти такие значения x_j и, следовательно, q_j, чтобы цена игры υ была наименьшей, то решение второй задачи сводится к нахождению таких неотрицательных значений x_j , $\left(j=\overline{1,n}\right)$, при которых $\sum_{j=1}^{n} x_j \to \max$, $\sum_{j=1}^{n} a_{ij} x_j \leq 1$.

Таким образом, поиск оптимальной смешанной стратегии первого игрока свелся к решению следующей задачи линейного программирования:

$$\varphi = \sum_{i=1}^{m} y_i (\min)$$

при ограничениях:

$$\sum_{i=1}^{m} a_{ij}\cdot y_i \geq 1 \left(j=\overline{1,n}\right)$$
$$y_i \geq 0 \left(i=\overline{1,m}\right).$$

Аналогично для второго игрока:

$$F = \sum_{j=1}^{n} x_i (\max)$$

при ограничениях:

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \le 1 \left(i=\overline{1,m}\right)$$
$$x_j \ge 0 \left(j=\overline{1,n}\right).$$

Найдя решения полученных задач линейного программирования, вычисляют цену игры и соответствующие вероятности по формулам:

$$\upsilon = \frac{1}{\sum_{i=1}^{m} y_i} \quad \text{или} \quad \upsilon = \frac{1}{\sum_{j=1}^{n} x_j}; \quad \frac{y_i}{\upsilon} = p_i \quad \left(i=\overline{1,m}\right), \quad \frac{x_j}{\upsilon} = q_j \quad \left(j=\overline{1,n}\right).$$

При решении произвольной конечной игры размера *m* x *n* рекомендуется придерживаться следующей схемы:

1. Исключить из платежной матрицы заведомо невыгодные стратегии по сравнению с другими стратегиями. Такими стратегиями для первого игрока (второго игрока) являются те, которым соответствуют строки (столбцы) с элементами, заведомо меньшими (большими) по сравнению с элементами других строк (столбцов).

2. Определить верхнюю и нижнюю цены игры и проверить, имеет ли игра седловую точку. Если седловая точка есть, то соответствующие ей стратегии игроков будут оптимальными, а цена совпадает с верхней (нижней) ценой.

3. Если седловая точка отсутствует, то решение следует искать в смешанных стратегиях.

Пример3. Найти решение игры, заданной матрицей

$$A = \begin{vmatrix} 1 & 5 & 4 \\ 3 & 2 & 6 \\ 8 & 3 & 4 \end{vmatrix}$$

Решение. Исследование в матричных играх начинается с нахождения седловой точки в чистых стратегиях. Если матричная игра имеет седловую точку в чистых стратегиях, то нахождением этой точки заканчивается решение игры и записывается ответ: цена игры, соответствующая найденной седловой точке и стратегии первого и второго игроков соответствующие этой цене игры. Если же игра не имеет седловой точки в чистых стратегиях, то найденные нижняя и верхняя чистые цены этой игры указывают, что первый игрок не должен надеяться на выигрыш больший, чем верхняя цена игры, и может быть уверен в получении выигрыша не меньше нижней цены игры.

Решение начинаем с исследования платежной матрицы игры. Прежде всего, проверим наличие седловой точки в данной матрице *A*. Для этого найдем минимальные элементы в каждой из строк (1, 2 и 3) и максимальные элементы в каждом из столбцов (8, 5 и 6).

$$A=\begin{pmatrix}1 & 5 & 4\\ 3 & 2 & 6\\ 8 & 3 & 4\end{pmatrix}\begin{matrix}\min\limits_j a_{ij}\\ \to 1\\ \to 2\\ \to 3\end{matrix}\Big\}\max_i \min_j a_{ij}=3$$

$$\max_i a_{ij}\downarrow \quad \downarrow \quad \downarrow$$

$$\underbrace{8 \quad 5 \quad 6}$$

$$\text{minmax}\, a_{ij}=5$$

(8.1)

Затем определим нижнюю цену игры $\alpha=\max\limits_i \min\limits_j a_{ij}=3$ и верхнюю цену игры $\beta=\min\limits_j \max\limits_i a_{ij}$. (рисунок 8.1).

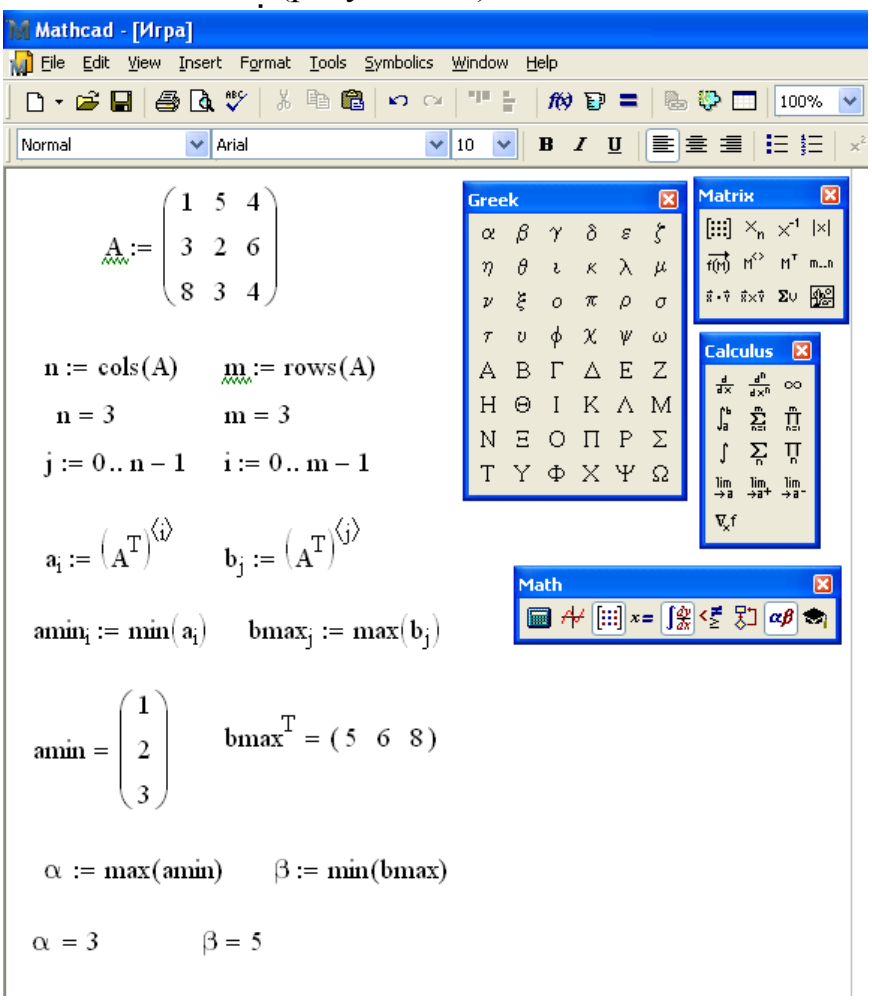


Рисунок 8.1 – Алгоритм поиска нижней и верхней цены игры в Mathcad

Так как нижняя цена игры $\alpha=3$, а верхняя цена игры $\beta=5$, т.е. игра не имеет седловой точки, а значит, нет решения в чистых стратегиях. Решением игры являются смешанные оптимальные стратегии, а цена игры ν заключена в пределах $3 \le \nu \le 5$.Предположим, первый игрок А

применяет свои чистые стратегии A_1, A_2 и A_3 с вероятностями p_1, p_2 и p_3 соответственно. Сумма вероятностей равна единице. Тогда, если второй игрок применит свои чистые стратегии B_1 или B_2 или B_3, то игрок А получит средний выигрыш, больше или равный цене игры, т. е. получим следующую систему неравенств:

$$\begin{cases} p_1+3p_2+8p_3 \geq v \left(\text{при стратегии } B_1\right) \\ 5p_1+2p_2+3p_3 \geq v \left(\text{при стратегии } B_2\right) \end{cases} . \quad (8.2)$$

Так как сумма вероятностей равна единице, получим следующее уравнение:

$$p_1+p_2+p_3=1 \quad (8.3)$$

Разделим все члены неравенств системы (1.9) и уравнение (1.10) на v и заменим полученные выражения соответственно на: $y_1=\frac{p_1}{v}, y_2=\frac{p_2}{v}, y_3=\frac{p_3}{v}$ Система (4.9) и уравнение (4.10) примет следующий вид:

$$\begin{cases} y_1+3y_2+8y_3 \geq 1 \\ 5y_1+2y_2+3y_3 \geq 1 \end{cases}$$

Так как первый игрок стремиться сделать свой выигрыш как можно больше, т. е. сделать значение цены игры максимальным, следовательно, сумма $y_1+y_2+y_3 \to \min$. Таким образом, получим следующую задачу линейного программирования:

Найти значения y_1, y_2, y_3 при которых целевая функция $\Psi=y_1+y_2+y_3$ примет минимальное значение:

$$\Psi=y_1+y_2+y_3 \to \min$$

при условии:

$$\begin{cases} y_1+3y_2+8y_3 \geq 1 \\ 5y_1+2y_2+3y_3 \geq 1 \end{cases}, \quad (8.4)$$

Найдя решения полученной задачи линейного программирования, вычислим цену игры и соответствующие вероятности по формулам:

$$v = \frac{1}{\Psi_{\min}} = \frac{1}{\sum_{i=1}^{m} y_i}; \quad \frac{y_i}{v} = p_i \qquad (i = \overline{1,m}).$$

Проведя аналогичные рассуждения для второго игрока, получим задачу линейного программирования двойственную к задаче (4.11):

Найти значения x_1, x_2, x_3 при которых целевая функция $F = x_1 + x_2 + x_3$ примет максимальное значение:

$$F = x_1 + x_2 + x_3 \to \max$$

при условии:

$$\begin{cases} x_1 + 5x_2 + 4x_3 \le 1 \\ 3x_1 + 2x_2 + 6x_3 \le 1 \\ \dots \end{cases} \qquad (8.5)$$

$$\frac{x_j}{v} = q_j \qquad (j = \overline{1,n}). \qquad \text{или} \qquad v = \frac{1}{\sum_{j=1}^{n} x_j}$$

Пример решения задачи в Excel для второго игрока приведен на рисунке 8.2 (алгоритм решения приведен ниже).

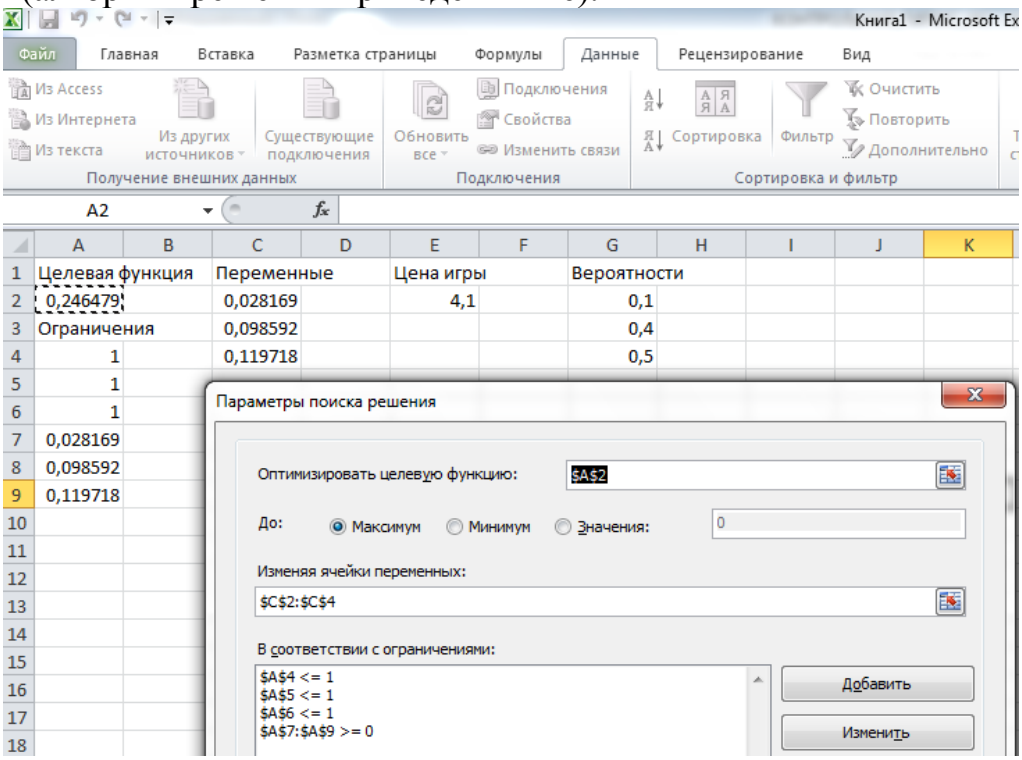


Рисунок 8.2 – Рабочий лист с результатами решения задачи поиска смешанных стратегий для второго игрока и заполненными полями окна диалога «Параметры поиска решения»

Аналогично находиться решение для первого игрока (рисунок 8.3).

Ответ: Оптимальная стратегия каждого игрока состоит в том, чтобы чередовать свои чистые стратегии случайным образом, т. е. первому игроку, чтобы выиграть не менее 4, надо чередовать применение первой и третьей стратегии, выбирая каждую с вероятностью 0,5, а вторую стратегию не применять. Второму игроку, чтобы не дать выиграть первому игроку больше, чем 4, необходимо в 4 раза реже применять свою первую чистую стратегию по сравнению со второй и в 5 раз реже по сравнению с третьей.

При решении задачи теории игр рекомендуется использовать следующий алгоритм:

1. Исключить из платежной матрицы игры заведомо невыгодные стратегии. Такими стратегиями для первого игрока (второго игрока) являются те, которым соответствуют строки (столбцы) с элементами, меньшими (большими) по сравнению с элементами других строк (столбцов).

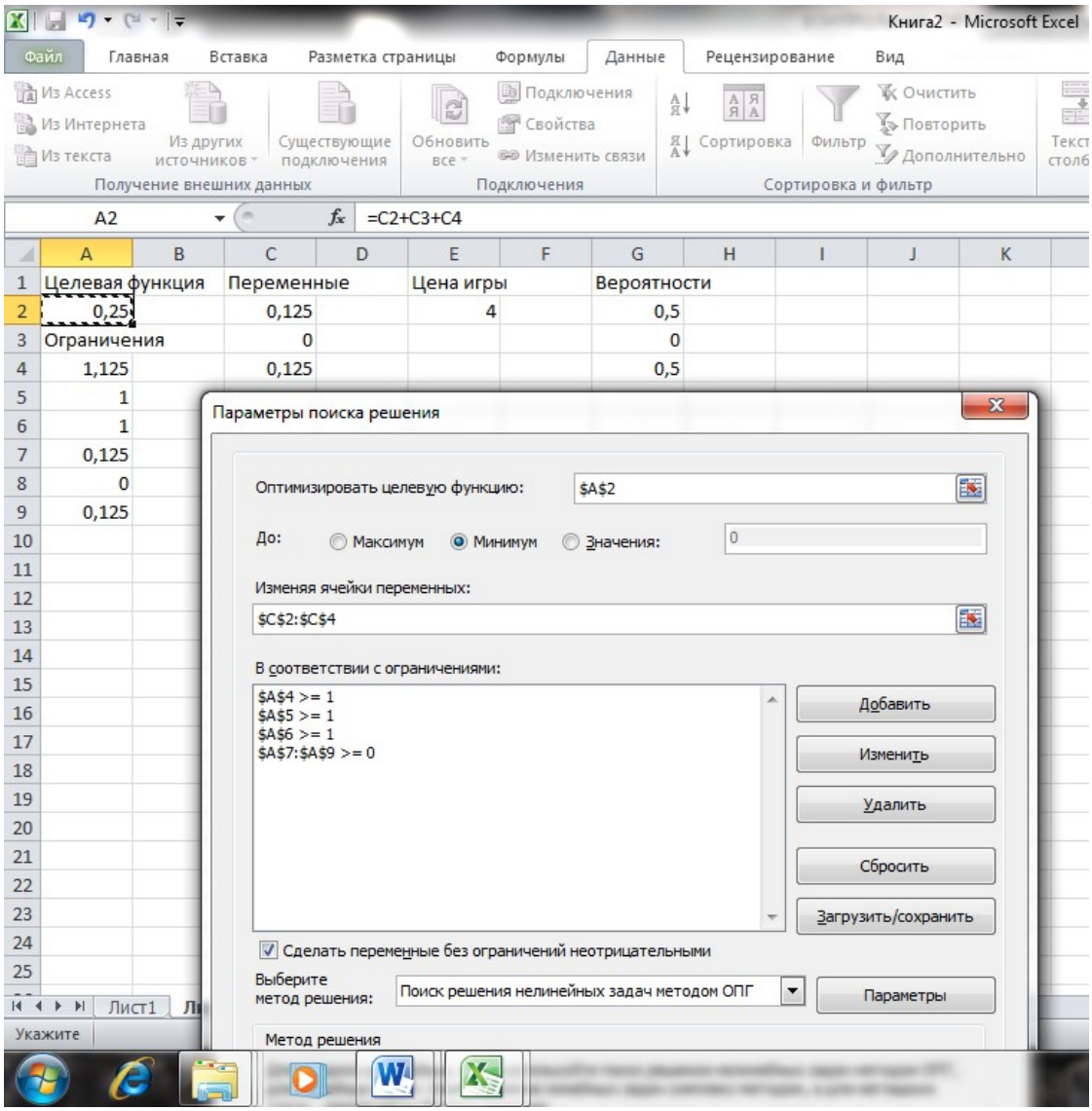


Рисунок 8.3 – Рабочий лист с результатами решения задачи для первого игрока и заполненными полями окна диалога «Параметры поиска решения»

2. Определить имеет ли игра седловую точку. Для этого необходимо найти верхнюю и нижнюю цену игры:

2.1. Если они равны, то игра имеет седловую точку, а значит решение в чистых стратегиях и сразу можно записать ответ, так как соответствующие ей стратегии игроков будут оптимальными.

2.2. Если верхняя и нижняя цены игры не равны, то необходимо искать решение игры в смешанных стратегиях путем сведения ее к задаче линейного программирования.

3. Процесс нахождения решения игры с использованием методов линейного программирования включает следующие этапы:

3.1. Составляют пару двойственных задач линейного программирования, эквивалентных данной матричной игре.

3.2. Определяют оптимальные планы пары двойственных задач.

3.3. Используя соотношение между планами пары двойственных задач, оптимальными стратегиями и ценой игры, находят решение игры.

Пример 3. Найти решение игры, определяемой матрицей:

$$A=\begin{pmatrix}0 & 1 & -1\\ 0 & -1 & 0\\ 1 & 0 & -1\end{pmatrix}.$$

Решение. При решении этой игры к каждому элементу матрицы A прибавим 1 и получим следующую матрицу:

$$A_1=\begin{pmatrix}1 & 2 & 0\\ 1 & 0 & 1\\ 2 & 1 & 0\end{pmatrix}.$$

Так как игра не имеет седловой точки, составим теперь пару взаимно-двойственных задач:

$$\begin{cases} y_1+y_2+2y_3\geq 1,\\ 2y_1+\quad\ \ y_3\geq 1,\\ \quad\ \ y_3\geq 1,\\ y_1,y_2,y_3\geq 0. \end{cases} \qquad y_1+y_2+y_3\to\min$$

$$\begin{cases} x_1+2x_2\quad\ \ \leq 1,\\ x_1\quad\ \ +x_3\leq 1,\\ 2x_1+x_2\quad\ \ \leq 1,\\ x_1,x_2,x_3\geq 0. \end{cases} \qquad x_1+x_2+x_3\to\max$$

Рассмотрим пример решения задачи с использованием инструмента «Поиск решения» электронной таблицы Excel.

Для определения оптимального плана второй задачи необходимо:

1. Записать целевую функцию в следующем виде:

$$f'(c_1,'c_2,c_3)=c_1+c_2+c_3\ ,$$

а ограничения – неравенства, следующим образом:

$$c_1+2*c_2\le 1$$
$$c_1+c_3\le 1$$
$$2*c_1+c_2\le 1$$
$$c_1\ge 0,\ c_2\ge 0, c_3\ge 0.$$

2. Выполнить следующие действия:

а) ввести в ячейку A_1 выражение для целевой функции: $=c_1+c_2+c_3$;

б) ввести ограничения в ячейки $A_3 : A_9$ и нулевые значения в ячейки $C_1 : C_3$;

в) выполнить команду **Сервис/Поиск решения** и заполнить параметры в окне диалога **Поиск решений**. Установить целевую ячейку A1, равной максимальному значению, в поле ввода **Изменяя ячейки:** ввести: c1:c3. Перейти в поле ввода **Ограничения:** и выполнить команду **Добавить**. Заполнить появившееся окно диалога, последовательно вводя ячейки, в которых находятся неравенства системы ограничений;

г) решить задачу, выбрав команду **Выполнить**, в появившемся окне **Результаты поиска решения** выбрать тип отчета **Результаты**;

д) вычислить значение цены игры. Для этого в ячейку e2 ввести формулу: =1/A1. Цена игры равна 0,67;

е) рассчитать значения вероятностей использования игроком своих стратегий. Для этого ввести в ячейки E4:E6 соответственно: =c1·e2; =c2·e2; =c3·e2. Вероятность использования игроком первой стратегии равна 0, второй стратегии – 0,33, третьей – 0,67.

Аналогично решая задачу, находим вероятность использования своих стратегий вторым игроком. Вероятность использования игроком первой стратегии равна 0,33, второй стратегии – 0,67, третьей – 0.

Пример 4. Построение матричной игры на основе симметричной пары двойственных задач

Для всякой матричной игры можно записать симметричную пару двойственных задач. Справедливо и обратное: для всякой симметричной пары двойственных задач можно записать матричную игру.

Пусть задана симметричная пара двойственных задач: прямая задача:

$F=CX,\ AX \le B,\ X \ge 0$, двойственная задача: $\Psi = BY,\ YA \ge C, Y \ge 0$. Тогда этой симметричной паре двойственных задач можно поставить в соответствие игру, определяемую матрицей

$$D=\begin{bmatrix} 0 & A & -B \\ -A^T & 0 & C^T \\ B^T & -C & 0 \end{bmatrix}, \qquad (8.6)$$

где индекс ^{T} означает операцию транспонирования.

Следует отметить, что если каждая матричная игра имеет оптимальные стратегии, то не всякая задача линейного программирования имеет решение.

Построить игру, определяемую следующей парой двойственных задач:

прямая задача:

$$F=2x_1+3x_2(\max)$$

при ограничениях

$$\begin{cases}2x_1+x_2\le 10\\ -x_1+3x_2\le 12\end{cases} \tag{8.7}$$

двойственная задача:

$$F=10y_1+12y_2(\min)$$

при ограничениях

$$\begin{cases}2y_1-y_2\ge 2\\ y_1+3y_2\ge 3\end{cases} \tag{8.8}$$

Решение. Запишем матрицы, элементами которых являются: коэффициенты при неизвестных левой части системы ограничений прямой и двойственной задач соответственно (матрицы A и A^{T}); коэффициенты при неизвестных в целевой функции прямой и двойственной задач соответственно (матрицы C и B^{T}); правые части системы ограничений прямой и двойственной задач соответственно (матрицы B и C^{T}):

$$A=\begin{bmatrix}2 & 1\\ -1 & 3\end{bmatrix};\ A^T=\begin{bmatrix}2 & -1\\ 1 & 3\end{bmatrix};\ B=\begin{bmatrix}10\\ 12\end{bmatrix}.$$

$$B^T=(10,12);\ C=(2,3);\ C^T=\begin{bmatrix}2\\ 3\end{bmatrix}.$$

Следовательно, исходной симметричной паре двойственных задач можно поставить в соответствие матричную игру, определяемую матрицей:

$$D=\begin{bmatrix}0 & 0 & 2 & 1 & -10\\ 0 & 0 & -1 & 3 & -12\\ -2 & 1 & 0 & 0 & 2\\ -1 & -3 & 0 & 0 & 3\\ 10 & 12 & -2 & -3 & 0\end{bmatrix}.$$

Аппаратура и материалы. Для выполнения лабораторной работы необходимо использовать следующее: аппаратное обеспечение: персональный компьютер и программное обеспечение: операционную систему Windows 10 и выше; Microsoft Office13 и выше, системы компьютерной математики Machcad и MATLAB R2017a и выше.

Указания по технике безопасности. Самостоятельно не производить установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

Выполните предложенные задания, предварительно разобрав примеры, приведенные выше.

Задание 8.1. Принятие решений в условиях неопределенности

1. Исследуйте платежную матрицу игры (таблица 8.1):
- определите нижнюю и верхнюю цены игры;
- определите минимаксные и максиминные стратегии;
- определите оптимальные решения игры в чистых стратегиях, если существует седловая точка.

2. Найдите решения игры, не имеющей седловую точку, путем сведения ее к задаче линейного программирования:
- запишите модели прямой и двойственной задачи;
- найдите решение обеих задач;
- проанализируйте правильность найденного решения с помощью двух теорем двойственности.

3. Сделайте вывод.

Таблица 8.1– Варианты второго задания

№ варианта	Платежная матрица игры	№ варианта	Платежная матрица игры
1.	$\begin{matrix}2&5&8\\\|7&2&8\|\\3&7&1\end{matrix}$	1.	$\begin{matrix}4&5&3\\\|6&8&4\|\\4&2&4\end{matrix}$
	$\begin{matrix}2&5&8\\\|7&2&8\|\\9&7&7\end{matrix}$		$\begin{matrix}4&5&8\\\|6&9&5\|\\5&2&1\end{matrix}$
2.	$\begin{matrix}3&5&6\\\|5&3&8\|\\3&7&1\end{matrix}$	2.	$\begin{matrix}2&5&8\\\|7&3&9\|\\3&5&1\end{matrix}$
	$\begin{matrix}3&3&8\\\|7&1&8\|\\3&3&2\end{matrix}$		$\begin{matrix}3&4&9\\\|3&2&8\|\\1&7&1\end{matrix}$

3.	2 5 9 |7 2 7| 3 3 1	3.	6 5 8 |7 1 8| 3 4 1
	3 5 4 |7 2 3| 5 7 5		4 1 7 |7 2 8| 3 6 2
4.	2 5 8 |7 3 5| 3 7 7	4.	6 5 3 |2 3 7| 3 7 1
	2 5 8 |4 3 5| 5 7 7		5 3 8 |6 2 7| 3 3 1
5.	5 5 8 |6 2 7| 3 4 1	5.	2 4 9 |4 3 5| 5 8 7
	5 8 3 |3 2 7| 3 7 1		2 5 8 |4 3 5| 9 7 3
6.	9 2 8 |3 2 4| 3 7 1	6.	2 5 8 |7 3 5| 5 6 4
	6 5 8 |7 2 8| 3 5 1		2 6 9 |7 3 5| 6 6 9
7.	3 5 9 |7 4 6| 3 7 2	7.	2 3 8 |4 4 5| 5 7 7
	2 5 8 |7 2 8| 3 7 1		2 5 8 |4 3 5| 5 6 9
8.	1 5 8 |7 4 3| 3 7 1	8.	3 5 8 |2 2 5| 1 7 1
	4 5 4 |7 4 3| 3 7 1		7 5 8 |4 3 5| 5 7 7

Задание 8.2. Постройте для симметричных пар двойственных задач определяемые ими матричные игры:

$$F=3x_1+4x_2+x_3 \to \max; \qquad F^*=12y_1+14y_2+16y_3 \to \max;$$

$$\begin{cases} 2x_1+3x_2-4x_3\le 12 \\ -x_1+x_2+x_3\le 14 \\ 3x_1+4x_2-x_3\le 16 \end{cases} \qquad \begin{cases} 2y_1-y_2+3y_3\ge 3 \\ 3y_1+y_2+4y_3\ge 4 \\ -4y_1+y_2-y_3\ge 1 \end{cases}$$

$$F=5x_1+7x_2+8x_3 \to \max; \qquad F^*=18y_1+16y_2+24y_3 \to \min;$$

$$\begin{cases} x_1+x_2+x_3\le 18 \\ 3x_1+2x_2+x_3\le 16 \\ 4x_1+3x_2+x_3\le 24 \end{cases} \qquad \begin{cases} y_1+3y_2+4y_3\ge 5 \\ y_1+2y_2+3y_3\ge 7 \\ 5y_1+y_2+y_3\ge 8 \end{cases}$$

Для исходной задачи составьте двойственную и постройте для полученной пары двойственных задач определяемые ими матричные игры:

$$F=2x_1+x_2+5x_3 \to \max; \qquad F=x_1+2x_2+3x_3 \to \max;$$

$$\begin{cases} x_1-x_2-x_3\le 4 \\ x_1-x_2+x_3\ge 5 \\ 2x_1-x_2+3x_3\ge 6 \end{cases} \qquad \begin{cases} 2x_1+2x_2-x_3\ge 2 \\ x_1-2x_2-4x_3\le -3 \\ x_1+x_2-2x_3\ge 6 \end{cases}$$

Содержание отчета и его форма

Подготовьте отчет, в котором опишите технологию решения задач линейного программирования средствами Excel, используя задания своего варианта. Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) условия выполненных заданий и их решение;
4) выводы;
5) ответы на контрольные вопросы.

Вопросы для защиты работы:

1. Что такое теория игр?

2. Что понимают под конфликтной ситуацией?
3. Что лежит в основе теории игр?
4. Перечислите основные понятия теории игр и дайте им определение.
5. По каким признакам проводят классификацию игр?
6. Назовите известные Вам классификации.
7. Какие основные способы описания и анализа игры Вы знаете?
8. Что такое платежная матрица игры?
9. Как определить по платежной матрице нижнюю и верхнюю цену игры?
10. С чего начинают решение игры?
11. В каком случае игра имеет решение в чистых стратегиях?
12. Что такое смешанная стратегия?
13. В чем заключается поиск оптимальной смешанной стратегии?

ЛАБОРАТОРНАЯ РАБОТА 9
Случайные процессы с дискретным состоянием. Уравнения Колмогорова для стационарного режима

Цель и содержание: Научиться составлять уравнения Колмогорова и выполнять анализ работы моделируемой системы.

Организационная форма занятий: решение проблемных задач, разбор конкретных ситуаций

Вопросы для обсуждения на лабораторном занятии: построение, решение и анализ моделей систем, описываемых уравнениями Колмогорова.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Изучите теоретический материал по данной теме.

Понятие случайного процесса

Строго говоря, случайные возмущения присущи любому процессу. Проще привести примеры случайного, чем «неслучайного» процесса. Даже, например, процесс хода часов (вроде бы это строгая выверенная работа – «работает как часы») подвержен случайным изменениям (уход вперед, отставание, остановка). Но до тех пор, пока эти возмущения несущественны, мало влияют на интересующие нас параметры, мы можем ими пренебречь и рассматривать процесс как детерминированный, неслучайный.

Пусть имеется некоторая система S (техническое устройство, группа таких устройств, технологическая система – станок, участок, цех, предприятие, отрасль промышленности и т.д.). В системе S протекает **случайный процесс**, если она с течением времени меняет свое состояние (переходит из одного состояния в другое), причем, заранее неизвестным случайным образом.

Примеры: 1. Система S – технологическая система (участок станков). Станки время от времени выходят из строя и ремонтируются. Процесс, протекающий в этой системе, случаен.

2. Система S – самолет, совершающий рейс на заданной высоте по определенному маршруту. Возмущающие факторы – метеоусловия, ошибки экипажа и т.д., последствия – «болтанка», нарушение графика полетов и т.д.

Марковский случайный процесс

Случайный процесс, протекающий в системе, называется **Марковским**, если для любого момента времени t_0 вероятностные характеристики процесса в будущем зависят только от его состояния в данный момент t_0 и не зависят от того, когда и как система пришла в это состояние.

Пусть в настоящий момент t_0 система находится в определенном состоянии S_0. Мы знаем характеристики состояния системы в настоящем $t_0 \to S_0$ и все, что было при $t < t_0$ (предысторию процесса). Можем ли мы предугадать (предсказать) будущее, т.е. что будет при $t > t_0$? В точности – нет, но какие-то вероятностные характеристики процесса в будущем найти можно. Например, вероятность того, что через некоторое время τ система S окажется в состоянии S_1 или останется в состоянии S_0 и т.д.

Пример. Система S – группа самолетов, участвующих в воздушном бою. Пусть x – количество «красных» самолетов, y – количество «синих» самолетов. К моменту времени t_0 количество сохранившихся (не сбитых) самолетов соответственно – x_0, y_0. Нас интересует вероятность того, что в момент времени $(t_0 + \tau)$ численный перевес будет на стороне «красных». Эта вероятность зависит от того, в каком состоянии находилась система в момент времени t_0, а не от того, когда и в какой последовательности погибали сбитые до момента t_0 самолеты.

На практике Марковские процессы в чистом виде обычно не встречаются. Но имеются процессы, для которых влиянием «предистории» можно пренебречь. И при изучении таких процессов можно применять Марковские модели (в теории массового обслуживания рассматриваются и не Марковские системы массового обслуживания, но математический аппарат, их описывающий, гораздо сложнее).

В исследовании операций большое значение имеют Марковские случайные процессы с дискретными состояниями и непрерывным временем.

Процесс называется **процессом с дискретным состоянием**, если его возможные состояния S_1, S_2, … можно заранее определить, и переход системы из состояния в состояние происходит «скачком», практически мгновенно.

Процесс называется **процессом с непрерывным временем**, если моменты возможных переходов из состояния в состояние не фиксированы заранее, а неопределенны, случайны и могут произойти в любой момент.

Далее рассматриваются только процессы с дискретным состоянием и непрерывным временем.

Пример1. Технологическая система (участок) S состоит из двух станков, каждый из которых в случайный момент времени может выйти из строя (отказать), после чего мгновенно начинается ремонт узла, тоже продолжающийся заранее неизвестное, случайное время. Возможны следующие состояния системы:

S_0 - оба станка исправны;

S_1 - первый станок ремонтируется, второй исправен;

S_2 - второй станок ремонтируется, первый исправен;

S_3 - оба станка ремонтируются.

Переходы системы S из состояния в состояние происходят практически мгновенно, в случайные моменты выхода из строя того или иного станка или окончания ремонта.

При анализе случайных процессов с дискретными состояниями удобно пользоваться геометрической схемой – **графом состояний**. Вершины графа – состояния системы. Дуги графа – возможные переходы из состояния в состояние. Для нашего примера граф состояний приведен на рисунке 9.1.

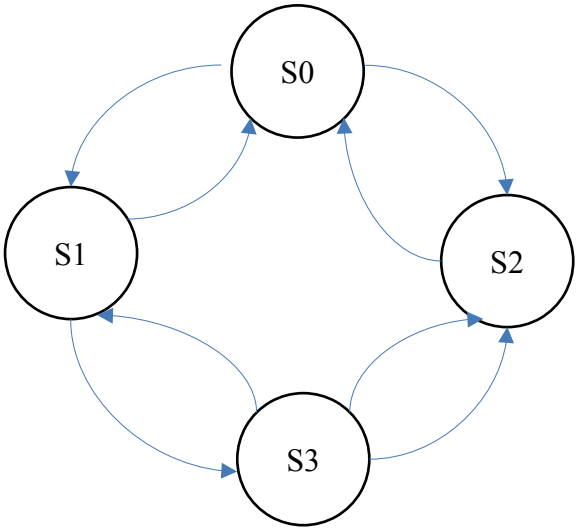


Рисунок 9.1 – Граф состояний системы

Примечание. Переход из состояния S_0 в S_3 на рисунке не обозначен, т.к. предполагается, что станки выходят из строя независимо друг от друга. Вероятностью одновременного выхода из строя обоих станков мы пренебрегаем.

Потоки событий

Поток событий – последовательность однородных событий, следующих одно за другим в какие-то случайные моменты времени.

В предыдущем примере – это поток отказов и поток восстановлений. Другие примеры: поток вызовов на телефонной станции, поток покупателей в магазине и т.д.

Поток событий можно наглядно изобразить рядом точек на оси времени $O\,t$ – рисунок 9.2.

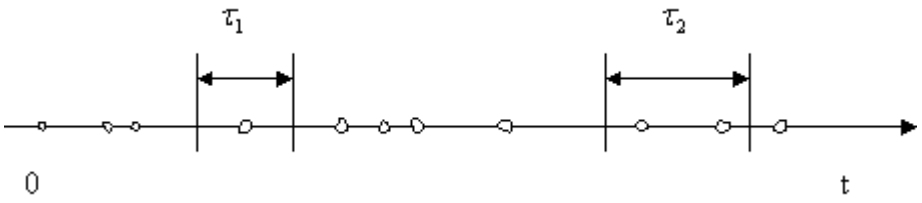


Рисунок 9.2. – Изображение потока событий на оси времени

Положение каждой точки случайно, и здесь изображена лишь какая-то одна реализация потока.

Интенсивность потока событий (λ) – это среднее число событий, приходящееся на единицу времени.

Рассмотрим некоторые свойства (виды) потоков событий.

Поток событий называется **стационарным**, если его вероятностные характеристики не зависят от времени.

В частности, интенсивность λ стационарного потока постоянна. Поток событий неизбежно имеет сгущения или разрежения, но они не носят закономерного характера, и среднее число событий, приходящееся на единицу времени, постоянно и от времени не зависит.

Поток событий называется **потоком без последствий**, если для любых двух непересекающихся участков времени τ_1 и τ_2 (см. рисунок 9.2.) число событий, попадающих на один из них, не зависит от того, сколько событий попало на другой. Другими словами, это означает, что события, образующие поток, появляются в те или иные моменты времени **независимо друг от друга** и вызваны каждое своими собственными причинами.

Поток событий называется **ординарным**, если события в нем появляются поодиночке, а не группами по нескольку сразу.

Поток событий называется **простейшим (или стационарным пуассоновским),** если он обладает сразу тремя свойствами:

1) стационарен,

2) ординарен,

3) не имеет последствий.

Простейший поток имеет наиболее простое математическое описание. Он играет среди потоков такую же особую роль, как и закон нормального распределения среди других законов распределения. А именно, при наложении достаточно большого числа независимых, стационарных и ординарных потоков (сравнимых между собой по интенсивности) получается поток, близкий к простейшему.

Для простейшего потока с интенсивностью λ интервал T между соседними событиями имеет так называемое **показательное (экспоненциальное) распределение** с плотностью

$$f(t) = \lambda e^{-\lambda t},$$

где λ - параметр показательного закона.

Для случайной величины T, имеющей показательное распределение, математическое ожидание m_T есть величина, обратная параметру, а среднее квадратичное отклонение σ_T равно математическому ожиданию

$$m_T = \sigma_T = 1/\lambda.$$

Уравнения Колмогорова для вероятностей состояний. Финальные вероятности состояний

Рассматривая Марковские процессы с дискретными состояниями и непрерывным временем, подразумевается, что все переходы системы *S* из состояния в состояние происходят под действием простейших потоков событий (потоков вызовов, потоков отказов, потоков восстановлений и т.д.). Если все потоки событий, переводящие систему *S* из состояния в состояние простейшие, то процесс, протекающий в системе, будет Марковским.

Итак, на систему, находящуюся в состоянии S_i, действует простейший поток событий. Как только появится первое событие этого потока, происходит «перескок» системы из состояния S_i в состояние S_j (на графе состояний по стрелке $S_i \to S_j$).

Для наглядности на графе состояний системы у каждой дуги проставляют интенсивности того потока событий, который переводит систему по данной дуге (стрелке). λ_{ij} - интенсивность потока событий, переводящий систему из состояния S_i в S_j. Такой граф называется **размеченным**. Для нашего примера размеченный граф приведен на рисунок 9.3.

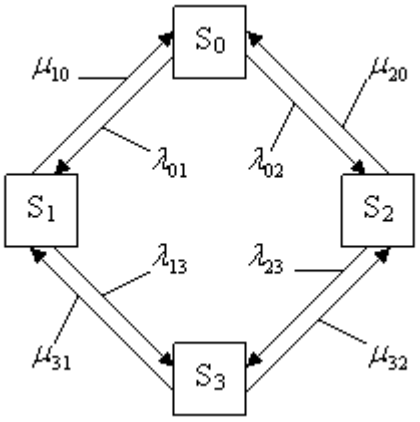


Рисунок 9.3. – Размеченный граф состояний системы

На этом рисунке λ_{ij} - интенсивности потока отказов; μ_{ij} - интенсивности потока восстановлений.

Предполагаем, что среднее время ремонта станка не зависит от того, ремонтируется ли один станок или оба сразу. Т.е. ремонтом каждого станка занят отдельный специалист.

Пусть система находится в состоянии S_0. В состояние S_1 ее переводит поток отказов первого станка. Его интенсивность равна

$$\lambda_{01} = 1/T_{ср.раб.1}, ед.времени^{-1},$$

где $T_{ср.раб1}$ – среднее время безотказной работы первого станка.

Из состояния S_1 в S_0 систему переводит поток «окончаний ремонтов» первого станка. Его интенсивность равна

$$\mu_{10} = 1/T_{ср.рем.1}, ед.времени^{-1}$$

где $T_{ср.рем}$ - среднее время ремонта первого станка.

Аналогично вычисляются интенсивности потоков событий, переводящих систему по всем дугам графа. Имея в своем распоряжении размеченный граф состояний системы, строится **математическая модель** данного процесса.

Пусть рассматриваемая система *S* имеет n - возможных состояний $S_1, S_2, \ldots, S_n$. Вероятность i - го состояния $p_i(t)$ - это вероятность того, что в момент времени t система будет находиться в состоянии S_i. Очевидно, что для любого момента времени сумма всех вероятностей состояний равна единице:

$$\sum_{i=1}^{n} p_i(t) = 1.$$

Для нахождения всех вероятностей состояний $p_i(t)$ как функций времени составляются и решаются **уравнения Колмогорова** – особого вида уравнения, в которых неизвестными функциями являются вероятности состояний. Правило составления этих уравнений приведем здесь без доказательств. Но прежде, чем его приводить, объясним понятие **финальной вероятности состояния**.

Что будет происходить с вероятностями состояний при $t \to \infty$? Будут ли $p_1(t), p_2(t), \ldots$ стремиться к каким-либо пределам? Если эти пределы существуют и не зависят от начального состояния системы, то они называются **финальными вероятностями состояний**.

$$\lim_{t \to \infty} p_i(t) = p_i, i = \overline{1, n},$$

где n - конечное число состояний системы.

Финальные вероятности состояний – это уже не переменные величины (функции времени), а постоянные числа. Очевидно, что

$$\sum_{i=1}^{n} p_i = 1.$$

Финальная вероятность состояния S_i – это по – существу среднее относительное время пребывания системы в этом состоянии.

Например, система *S* имеет три состояния S_1, S_2 и S_3. Их финальные вероятности равны соответственно 0,2; 0,3 и 0,5. Это значит, что система в предельном стационарном состоянии в среднем 2/10 времени проводит в состоянии S_1, 3/10 – в состоянии S_2 и 5/10 – в состоянии S_3.

Правило составления системы уравнений Колмогорова: в каждом уравнении системы **в левой его части** стоит финальная вероятность данного состояния p_i, умноженная на суммарную интенсивность всех потоков, **ведущих из данного состояния**, а **в правой его части** – сумма произведений интенсивностей всех потоков,

входящих в i- е состояние, на вероятности тех состояний, из которых эти потоки исходят.

Пользуясь этим правилом, напишем систему уравнений **для нашего примера**:

$$\begin{cases}(\lambda_{01}+\lambda_{02})p_0 = \mu_{10}p_1 + \mu_{20}p_2 \\ (\mu_{10}+\lambda_{13})p_1 = \lambda_{01}p_0 + \mu_{31}p_3 \\ (\mu_{20}+\lambda_{23})p_2 = \lambda_{02}p_0 + \mu_{32}p_3 \\ (\mu_{31}+\mu_{32})p_3 = \lambda_{13}p_1 + \lambda_{23}p_2\end{cases}$$

Эту систему четырех уравнений с четырьмя неизвестными p_0, p_1, p_2, p_3, казалось бы, можно вполне решить. Но эти уравнения однородны (не имеют свободного члена), и, значит, определяют неизвестные только с точностью до произвольного множителя. Однако можно воспользоваться нормировочным условием

$$p_0 + p_1 + p_2 + p_3 = 1$$

и с его помощью решить систему. При этом одно (любое) из уравнений можно отбросить (оно вытекает как следствие из остальных).

Продолжение примера 1. Пусть значения интенсивностей потоков равны:

$$\lambda_{01} = \lambda_{23} = 1; \lambda_{13} = \lambda_{02} = 2; \mu_{10} = \mu_{32} = 2; \mu_{20} = \mu_{31} = 3.$$

Четвертое уравнение отбрасываем, добавляя вместо него нормировочное условие:

$$\begin{cases}3p_0 = 2p_1 + 3p_2 \\ 4p_1 = p_0 + 3p_3 \\ 4p_2 = 2p_0 + 2p_3 \\ p_0 + p_1 + p_2 + p_3 = 1\end{cases}$$

$$p_0 = 6/15 = 0{,}4;\ p_1 = 3/15 = 0{,}2;\ p_2 = 4/15 \cong 0{,}27;\ p_3 = 2/15 \cong 0{,}13.$$

Т.е. в предельном, стационарном режиме система S в среднем 40 % времени будет проводить в состоянии S_0 (оба станка исправны), 20 % - в состоянии S_1 (первый станок ремонтируется, второй работает), 27 % - в состоянии S_2 (второй станок ремонтируется, первый работает), 13% - в состоянии S_3 (оба станка ремонтируются). Знание этих финальных вероятностей может помочь оценить среднюю эффективность работы системы и загрузку ремонтных органов.

Пусть система S в состоянии S_0 (полностью исправна) приносит в единицу времени доход 8 условных единиц, в состоянии S_1 – доход 3 условные единицы, в состоянии S_2 – доход 5 условных единиц, в состоянии S_3 – не приносит дохода. Тогда в предельном, стационарном режиме средний доход в единицу времени будет равен $W = 0{,}4*8 + 0{,}2*3 + 0{,}27*5 = 5{,}15$ условных единиц.

Станок 1 ремонтируется долю времени, равную $p_1 + p_3 = 0,2 + 0,13 = 0,3$. Станок 2 ремонтируется долю времени, равную $p_2 + p_3 = 0,27 + 0,13 = 0,4$. Возникает **задача оптимизации**. Пусть мы можем уменьшить среднее время ремонта первого или второго станка (или обоих), но это нам обойдется в определенную сумму. Спрашивается, окупит ли увеличение дохода, связанное с ускорением ремонта, повышенные расходы на ремонт? Нужно будет решить систему четырех уравнений с четырьмя неизвестными.

Системы массового обслуживания бывают двух видов: системы с отказами (потерями) и системы с очередью. Существуют системы с ограниченной очередью и неограниченной очередью. Ограничения могут быть по числу заявок, по времени обслуживания, по времени работы системы. Кроме того, учитывается дисциплина обслуживания. Заявки могут обслуживаться в порядке поступления их в систему, а могут рассматриваться в системе обслуживания с приоритетом. Различают приоритет абсолютный, относительный и смешанный.

Рассмотрим *n*- канальную систему с отказами. Граф состояний этой системы имеет вид (рисунок 9.1).

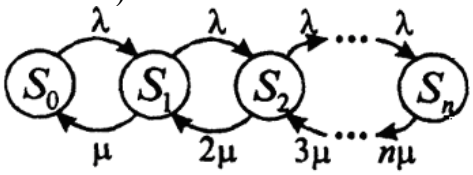


Рисунок 9.4 – Граф состояний n- канальной системы с отказами

Система может находиться в следующих состояниях:

S_0 – все каналы свободны;

S_1 – занят 1 канал, остальные свободны;

S_2 –заняты 2 канала, остальные свободны;

S_n – заняты все *n* каналов;

λ – интенсивность потока заявок;

μ – интенсивность потока обслуживания.

Пусть система находится в стационарном режиме. Пользуясь графом состояний, составим систему уравнений Колмогорова.

$$\begin{cases} 0 = -\lambda P_0 + \mu P_1 \\ \dots \\ 0 = -(\lambda + k\mu) P_k + \lambda P_{k-1} + (k+1) * \mu * P_{k+1}, k = 1,2,\dots,n-1. \\ \dots \\ 0 = -n\mu P_n + \lambda P_n + \lambda P_{n-1}. \end{cases} \quad (9.1)$$

Условия нормировки вероятностей имеет вид:

$$\sum_{k=0}^{n} P_k = 1.$$

Решая полученную систему уравнений, находим:

$$P_0=\left[\sum_{k=1}^{n}\frac{(\lambda/\mu)^k}{k!}\right]^{-1}(7.1)u \quad P_k=\frac{(\lambda/\mu)^k}{k!}\cdot P_0\,, \tag{9.2}$$

где $k = 0,1,\ldots, n$.

Заявка получает отказ, если система находится в состоянии S_n, следовательно, вероятность того, что заявка будет обслуживаться равна:

$$Q=1-P_n. \tag{9.3}$$

Интенсивность потока обслуженных заявок равна:

$$j=q\lambda, \tag{9.4}$$

а среднее число занятых каналов:

$$N=\frac{I}{\mu}. \tag{9.5}$$

Пример решения системы линейных уравнений в Excel

$$\begin{cases}3x_1+2x_2-5x_3=6,\\ x_1+6x_3=1,\\ \ldots\end{cases} \tag{9.6}$$

$$A=\begin{vmatrix}3 & 2 & -5\\ 1 & 0 & 6\\ 1 & -4 & 7\end{vmatrix};\quad X=\begin{vmatrix}x_1 & x_2 & x_3\end{vmatrix};\quad B=\begin{vmatrix}6\\ 1\\ 5\end{vmatrix},$$

т. е. в матричном виде система уравнений имеет вид: $A\cdot x=B$, решая которую получаем: $x=A^{-1}\cdot B$ где A^{-1} – обратная матрица. Чтобы обратить матрицу в Excel, необходимо выполнить следующие действия:

1. Ввести исходную матрицу в ячейки рабочего листа.
2. Указать на рабочем листе место, где будет находиться обратная матрица, выделив блок ячеек такого же размера, как исходная матрица.
3. Вызвать мастер функций.
4. В появившемся окне диалога выбрать функцию **МОБР** из категории **Математические** нажать кнопку **Далее** или клавишу **Enter**.
5. В появившемся окне диалога указать диапазон ячеек исходной матрицы и нажать кнопку **ОК** или клавишу **Enter**.
6. Переместить указатель мыши в строку формул и нажать левую кнопку мыши. Затем нажать одновременно три клавиши **Ctrl+Shift+Enter**.
7. Проверить правильность полученного результата, для этого умножить исходную матрицу на обратную матрицу с помощью функции **МУМНОЖ**. Если все сделано верно, должна получиться единичная матрица, это значит, что с помощью функции **МОБР** обратная матрица найдена правильно.
8. Умножение матриц производится следующим образом:

- выделяется место, где должна находиться результирующая матрица;
- вызывается мастер функций;
- выбирается функция **МУМНОЖ** из категории **Математические**;
- в появившемся окне диалога в поле: **Массив 1** вводится адрес блока ячеек, содержащего исходную матрицу, а в поле: **Массив 2** – адрес блока ячеек, содержащего обратную матрицу;
- затем выполняются действия, описанные в пункте 6.

Аппаратура и материалы. Для выполнения лабораторной работы необходимо использовать следующее: аппаратное обеспечение: персональный компьютер и программное обеспечение: операционную систему Windows 10 и выше; Microsoft Office13 и выше, систему компьютерной математики MATLAB R2017a и выше.

Указания по технике безопасности. Самостоятельно не производить установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

Выполните предложенные задания.

Задание 9.1. Граф состояний системы**.**

1. Изобразите граф состояний системы, состоящей из двух устройств, учитывая, что устройства могут выходить из строя, причем отказавшее устройство сразу начинают ремонтировать.
2. Составьте уравнения Колмогорова для каждого состояния системы, учитывая, что режим стационарный, интенсивности потоков отказов устройств и интенсивности потоков окончаний ремонта известны.
3. Замените последнее уравнение полученной системы уравнений условием нормировки.
4. Определите вероятности для каждого состояния системы, изображенной на графе, решив систему уравнений Колмогорова с помощью Excel. Данные для расчета представлены в таблице 9.1. Вариант задания определяется номером персонального компьютера, за которым работает студент в компьютерном классе. Формулировка задания для каждого варианта приведена в таблице 9.1.

Таблица 9.1 – Интенсивности потоков отказов устройств и потоков окончаний ремонта

Номер варианта	Интенсивности потоков отказа устройств	Интенсивности потоков окончания ремонта
1.	$\lambda_1=1$	$\mu_1=2$
	$\lambda_2=2$	$\mu_2=1$
	$\lambda_1=2$	$\mu_1=2$

Номер варианта	Интенсивности потоков отказа устройств	Интенсивности потоков окончания ремонта
21.	$\lambda_1=0,5$	$\mu_1=2$
	$\lambda_2=2$	$\mu_2=1$
	$\lambda_1=2$	$\mu_1=2$

3.	$\lambda_2=2$	$\mu_2=3$	23.	$\lambda_2=1,5$	$\mu_2=1$
5.	$\lambda_1=2$	$\mu_1=1$	25.	$\lambda_1=2$	$\mu_1=0,5$
	$\lambda_2=4$	$\mu_2=3$		$\lambda_2=1,5$	$\mu_2=2,5$
7.	$\lambda_1=1$	$\mu_1=4$	27.	$\lambda_1=1.5$	$\mu_1=2$
	$\lambda_2=3$	$\mu_2=1$		$\lambda_2=3$	$\mu_2=1$
9.	$\lambda_1=0,5$	$\mu_1=1,5$	29.	$\lambda_1=0,5$	$\mu_1=1,5$
	$\lambda_2=2$	$\mu_2=1$		$\lambda_2=1$	$\mu_2=0,5$
11.	$\lambda_1=1$	$\mu_1=0,5$	31.	$\lambda_1=1$	$\mu_1=0,5$
	$\lambda_2=0,5$	$\mu_2=1$		$\lambda_2=0,5$	$\mu_2=2$
13.	$\lambda_1=2,5$	$\mu_1=0,5$	33.	$\lambda_1=2,5$	$\mu_1=0,5$
	$\lambda_2=1,5$	$\mu_2=2$		$\lambda_2=2$	$\mu_2=1,5$
15.	$\lambda_1=1,5$	$\mu_1=2,5$	35.	$\lambda_1=1,5$	$\mu_1=2,5$
	$\lambda_2=3$	$\mu_2=1$		$\lambda_2=3$	$\mu_2=0,5$
17.	$\lambda_1=2$	$\mu_1=2$	37.	$\lambda_1=1$	$\mu_1=1$
	$\lambda_2=2,5$	$\mu_2=1,5$		$\lambda_2=2,5$	$\mu_2=1,5$
19.	$\lambda_1=3$	$\mu_1=0,5$	39.	$\lambda_1=3$	$\mu_1=1,5$
	$\lambda_2=1,5$	$\mu_2=3$		$\lambda_2=1,5$	$\mu_2=2$

5. Для решения системы используйте функции **МОБР** (возвращает обратную матрицу) и **МУМНЖ** (возвращает произведение матриц) из категории **Математические.**

6. Для вывода всех значений в вычисленных матрицах одновременно нажмите клавиши **Ctrl+Shift+Enter.**

Задание 9.2. Моделирование работы системы

1. Рассчитайте прибыль, которую приносит система, если известно, что первое устройство приносит доход 10 у.е., а второе 15 у.е.

2. Установите, к какому из устройств выгоднее всего применить рационализацию, позволяющую снизить в два раза время ремонта любого из устройств.

3. Подсчитайте доход в каждом из вариантов.

4. Сделайте вывод.

5. Оформите результаты расчетов.

Задание 9.3. Решить задачу.

На телефонную станцию поступают в среднем 1,5 заявки в минуту, среднее время обслуживания одной заявки 2 минуты. Станция имеет три канала обслуживания. Найти вероятность обслуживания поступившей заявки, среднее число занятых каналов, средний процент заявок, получивших отказ. Увеличить число каналов обслуживания и рассмотреть случаи, когда $n = 4, 5, 6$. Сделать вывод.

1. Нарисуйте граф состояний для этой задачи, составьте уравнение Колмогорова, пользуясь графом состояний. Решите задачу аналитически.

2. Используя формулы (9.1), (9.2), (9.3), (9.4), (9.5), напишите программу для решения задачи. Предусмотрите возможность изменений числа каналов обслуживания, интенсивности потока поступления заявок

и потока обслуживания.

3. Проверьте правильность работы вашей программы, используя данные п. 1).

4. Увеличить число каналов обслуживания и рассмотреть случаи, когда $n = 4, 5, 6$. Сделать вывод.

Содержание отчета и его форма

Подготовьте отчет, в котором опишите технологию решения систем линейных уравнений в Excel, используя задания своего варианта. Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) условия выполненных заданий и их решение;
4) выводы;
5) ответы на контрольные вопросы.

Вопросы для защиты работы:

1. Что такое граф состояний?
2. Какой режим является стационарным?
3. Как построить граф состояний системы?
4. Как составляются уравнения Колмогорова для состояний анализируемой системы?
5. Как решить систему уравнений с помощью инструментального средства **Поиск решения**?
6. Изобразите граф состояний одноканальной системы обслуживания с отказами.
7. Запишите уравнения Колмогорова для описания работы одноканальной системы обслуживания с отказами.
8. Под действием чего происходит уменьшение числа заявок в системе?
9. Чем определяется интенсивность потока обслуживания?
10. Изобразите граф состояний многоканальной системы обслуживания с отказами.
11. Запишите уравнения Колмогорова для описания работы многоканальной системы обслуживания с отказами.
12. Если одна ЭВМ обеспечивает интенсивность потока обслуживания μ, то какую интенсивность обслуживания обеспечат три ЭВМ, работающие одновременно?
13. Какими математическими отношениями описываются финальные вероятности состояний системы в результате решения системы уравнений Колмогорова?
14. Перечислите основные показатели эффективности многоканальной системы обслуживания с отказами.
15. Что требуется при заданной интенсивности потока заявок увеличивать для уменьшения времени пребывания задания в системе, а

значит, и в очереди?

16. Что требуется при заданной интенсивности потока заявок уменьшать для уменьшения времени пребывания задания в системе, а значит, и в очереди?

17. В каких случаях моделирование работы вычислительной системы проводится с помощью метода статистических испытаний (метода Монте-Карло)?

ЛАБОРАТОРНАЯ РАБОТА 10
Моделирование работы системы массового обслуживания

Цель и содержание: Научиться моделировать работу реальной системы массового обслуживания.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Изучите теоретический материал по данной теме, используя литературу [1,2] и материал, приведенный ниже.

Пусть число мест в очереди равно m . Если все места в очереди заняты, заявка получает отказ. Система имеет один канал обслуживания. Граф состояний этой системы имеет вид (рисунок 10.1):

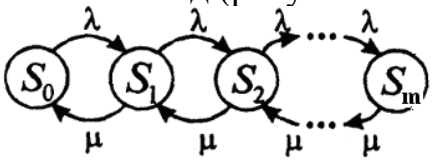


Рисунок 10.1 – Граф состояний многоканальной системы обслуживания с ограниченной очередью

S_0 – канал свободен;
S_1– канал занят;
S_2 – канал занят, (k-1) заявок в очереди;
S_m – канал занят, m заявок в очереди;
λ – интенсивность потока заявок;
μ – интенсивность потока обслуживания.

Уравнения Колмогорова имеют вид:

$$\begin{cases} 0=-\lambda\cdot P_0+\mu\cdot P_1 \\ 0=-(\lambda+\mu)\cdot P_1+\lambda\cdot P_0+\mu\cdot P_2 \\ \dots \\ 0=-(\lambda+\mu)\cdot P_k+\lambda\cdot P_{k-1}+\mu\cdot P_{r+1}, k=1,2,\dots,m+1 \\ \dots \\ 0=-(\lambda+\mu)\cdot P_{m+1}+\lambda\cdot P_m . \end{cases} \quad (10.1)$$

Условия нормировки вероятностей имеет вид: $\sum_{k=0}^{m+1} p_k=1$.

Решая полученную систему уравнений, находим:

$$P_0=\left[\sum_{i=0}^{m+1}\left(\frac{\lambda}{\mu}\right)^i\right]^{-1} . \quad (10.2)$$

и

$$P_k = \left(\frac{\lambda}{\mu}\right)^k \cdot P_0 \quad , \tag{10.3}$$

где $k = 0,1,\ldots,n.$

Вероятность отказа $p_{m+1};$ среднее число заявок в очереди равно:

$$N = \sum_{k=1}^{m} k \cdot P_{k+1} \quad . \tag{10.4}$$

Среднее время ожидания:

$$\frac{N}{\lambda} \quad . \tag{10.5}$$

Аппаратура и материалы. Для выполнения лабораторной работы необходимо использовать следующее: аппаратное обеспечение: персональный компьютер и программное обеспечение: операционную систему Windows 10 и выше; Microsoft Office13 и выше, системы компьютерной математики Machcad и MATLAB R2017a и выше.

Указания по технике безопасности. Самостоятельно не производить установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

Выполните предложенные задания.

Задание 1. Решите задачу. Пусть в систему прибывает в минуту в среднем одна заявка, длительность обслуживания заявки составляет две минуты. Число мест в очереди равно трем. Найти вероятность отказа, среднее время ожидания.

1. Нарисуйте граф состояний для этой задачи, составьте уравнение Колмогорова, пользуясь графом состояний. Решите задачу аналитически.
2. Напишите программу, используя формулы (10.1), (10.2), (10.3), (10.4) и (10.5) и варьируя число мест в очереди.
3. Проверьте правильность работы вашей программы, используя данные п. 1).
4. Увеличить число мест в очереди. Увеличивая m , убедиться, что при больших m вероятность отказа стабилизируется, становясь равной

$$\left(\frac{\lambda}{\mu} - 1\right) / \left(\frac{\lambda}{\mu}\right) \quad .$$

Сделать вывод о том, каким образом можно существенно снизить вероятность отказа.

Задание 2. Решите задачу**.** Бригада из r операторов обслуживает n

однотипных ЭВМ $(r \le n)$. Каждая из этих ЭВМ в случайные моменты времени может потребовать к себе внимания оператора. ЭВМ выходят из рабочего состояния независимо друг от друга; вероятность выхода из рабочего состояния равна λ. Вероятность восстановления рабочего состояния равна μ. Каждый оператор может одновременно восстанавливать только одну ЭВМ, каждая ЭВМ восстанавливается только одним оператором. Учитывая, что при $1 \le k \le r\left(\rho = \frac{\lambda}{\mu}\right)$, $P_k = \frac{n!}{k! \cdot (n-k)!} \cdot \rho_k \cdot P_0$, при $r \le k \le n$, $P_k = \frac{n!}{r^{n-k} \cdot r! \cdot (n-k)!} \cdot p^k \cdot P_0$, и $P_0 = \left[\sum_{k=0}^{r} \frac{n!}{k! \cdot (n-k)!} \cdot \rho_k + \sum_{k=r+1}^{n} \frac{n!}{r! \cdot r^{n-k} \cdot (n-k)!} \cdot \rho_k \right]^{-1}$.

Найти вероятность того, что в установившемся процессе обслуживания в данный момент будет простаивать то или иное число ЭВМ.

1. Пусть обслуживание 8 ЭВМ поручено двум операторам. Как рациональнее организовать работу: поручить ли все ЭВМ обоим операторам, чтобы по мере надобности к ЭВМ подходил один из свободных операторов, или же каждому оператору поручить по четыре определенных ЭВМ. Вычисления произвести в предположении $\rho = 0.2$.

2. Результаты расчетов привести в таблицах 10.1 и 10.2.

Таблица 10.1 – $n = 8, r = 2$

Число неработающих ЭВМ	Число ЭВМ, ожидающих обслуживания	Число свободных операторов	P_k
0	0	2	
1	0	1	
2	0	0	
3	1	0	
4	2	0	
5	3	0	
6	4	0	
7	5	0	
8	6	0	

Таблица 10.2 – $n = 4, r = 1$

Число неработающих ЭВМ	Число ЭВМ, ожидающих обслуживания	Число свободных операторов	P_k
0	0	1	
1	0	0	
2	1	0	
3	2	0	
4	3	0	

Среднее число ЭВМ, простаивающих в данный момент по той причине, что операторы заняты другими ЭВМ, равно $\sum_{k=2}^{n}(k-2)\cdot P_k$. Общее время простоя ЭВМ (восстановление и ожидание) равно: $\sum_{k=2}^{n} k\cdot P_k$

3. Сделать вывод.

Содержание отчета и его форма

Подготовьте отчет, в котором приведите технологию выполнения заданий.

Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) формулировку задания и технологию его выполнения;
4) ответы на контрольные вопросы;
5) приложение – файлы выполненных заданий.

Вопросы для защиты работы:

1. Изобразите граф состояний системы обслуживания вычислительных задач с очередью.
2. Какого типа бывают ограничения, наложенные на ожидание?
3. Что такое «дисциплина очереди»?
4. Что понимают под приоритетным обслуживанием?
5. Какие виды приоритета Вам известны?
6. Охарактеризуйте каждый вид приоритета и приведите пример.
7. Запишите уравнения Колмогорова для описания работы системы обслуживания вычислительных задач с очередью.
8. Запишите уравнения Колмогорова для описания работы системы обслуживания вычислительных задач с очередью.
9. Какими математическими отношениями описываются финальные вероятности состояний системы в результате решения системы уравнений Колмогорова?
10. Перечислите основные показатели эффективности системы обслуживания с очередью.
11. Запишите условие конечности длины очереди.
12. При каком условии очередь растет до бесконечности?

ЛАБОРАТОРНАЯ РАБОТА 11
Моделирование поведения динамической системы

Цель и содержание: изучить методику исследования поведения динамических систем, приобрести навыки реализации таких моделей в системе компьютерной математики MATLAB.

Организационная форма занятий: решение проблемных задач, разбор конкретных ситуаций

Вопросы для обсуждения на лабораторном занятии: исследование поведения динамической системы.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Распространенным способом описания поведения динамической системы является система дифференциальных или интегро-дифференциальных уравнений.

Реализацию таких математических моделей в Simulink рассмотрим на ряде примеров.

Пример 1. Модель физического маятника, находящегося под воздействием экспоненциально-затухающего косинусоидального возмущения.

Уравнение движения такого маятника имеет вид:

$$\begin{aligned} &\frac{d^2y(t)}{dt^2}+a_1\frac{dy(t)}{dt}+a_2\,y(t)=a_3e^{-a_4t}\cdot\cos a_5t,\\ &y(t)\big|_{t=0}=y_0,\\ &\left.\frac{dy(t)}{dt}\right|_{t=0}=y_0^{'}, \end{aligned} \qquad (11.1)$$

Выбрав числовые значения параметров, например: $a_1=a_2=0{,}1, a_3=-5, a_4=1, a_5=0{,}1$, получим следующее уравнение:

$$\begin{aligned} y^{'}+0{,}1\,y^{'}+0{,}1\,y&=-5\,e^{-t}\cdot\cos(0{,}1\,t),\\ y(0)&=-1{,}5,\\ y^{'}(0)&=2. \end{aligned} \qquad (11.2)$$

Структурная схема модели будет иметь вид, показанный на рисунке 11.1.

Результаты работы модели показан на экране виртуального осциллографа (рисунок 11.2), а параметрический график зависимости производной сигнала от сигнала (фазовый портрет маятника) изображен на рисунке 11.3.

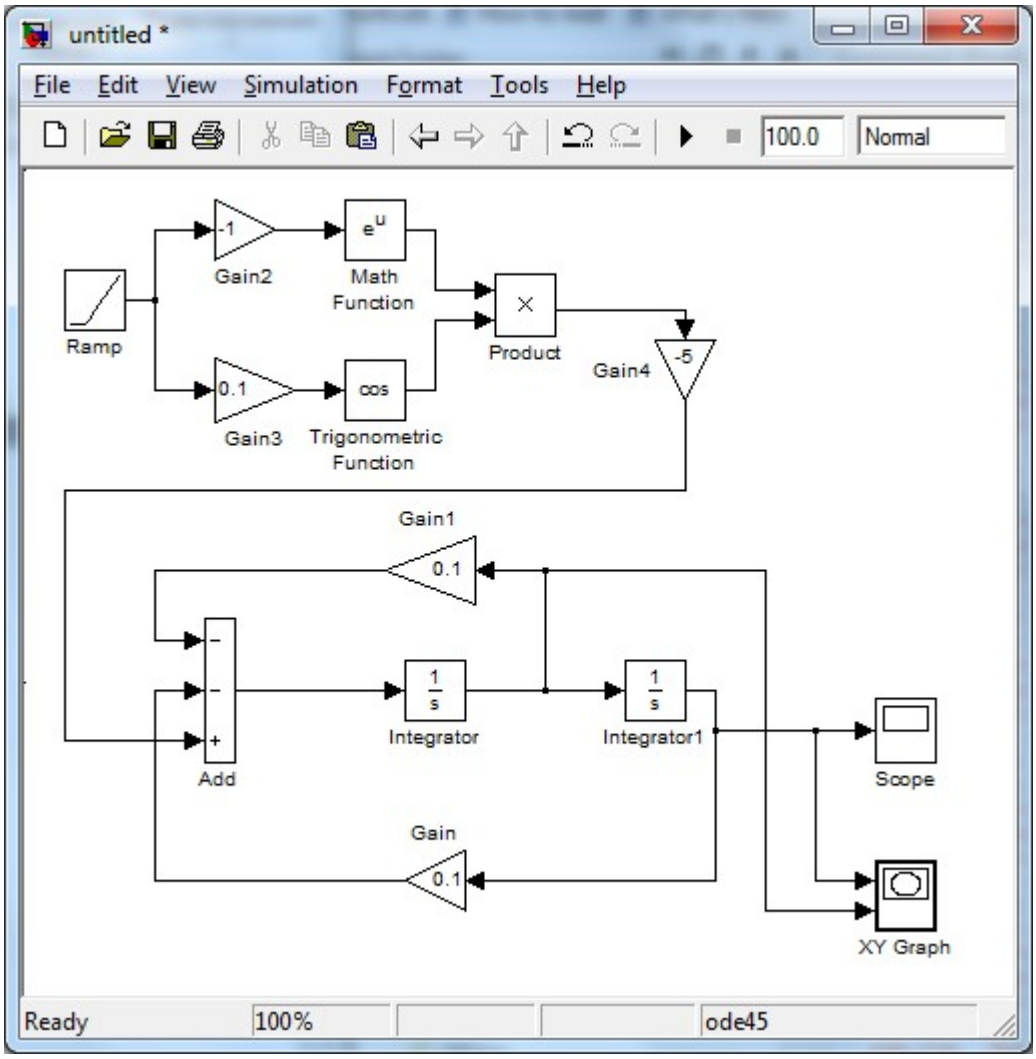


Рисунок 11.1 – Модель физического маятника

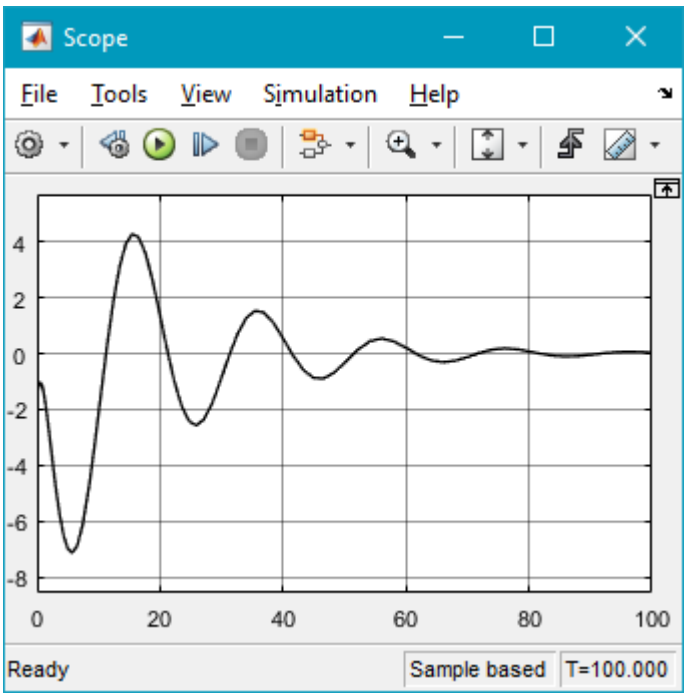


Рисунок 11.2 – Движение физического маятника

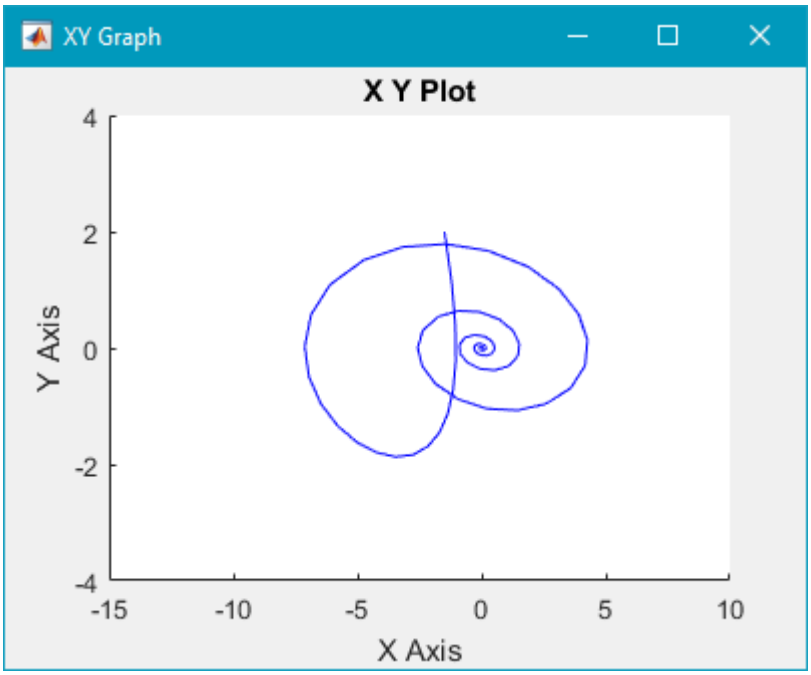


Рисунок 11.3 – Фазовый портрет физического маятника

Пример 2. Модель динамической системы, описываемой дифференциальным уравнением 3-го порядка.

Задано дифференциальное уравнение:

$$\frac{d^3y(t)}{dt^3}+2{,}5\frac{d^2y(t)}{dt^2}+6\frac{dy(t)}{dt}+2{,}5\,y(t)=e^{-t}, \qquad (11.3)$$

$$y(0)=1;\ y'(0)=-1;\ y''(0)=2.$$

Структурная схема модели динамической системы, построенная по обычным правилам аналоговой вычислительной техники, приведена на рисунке 11.4.

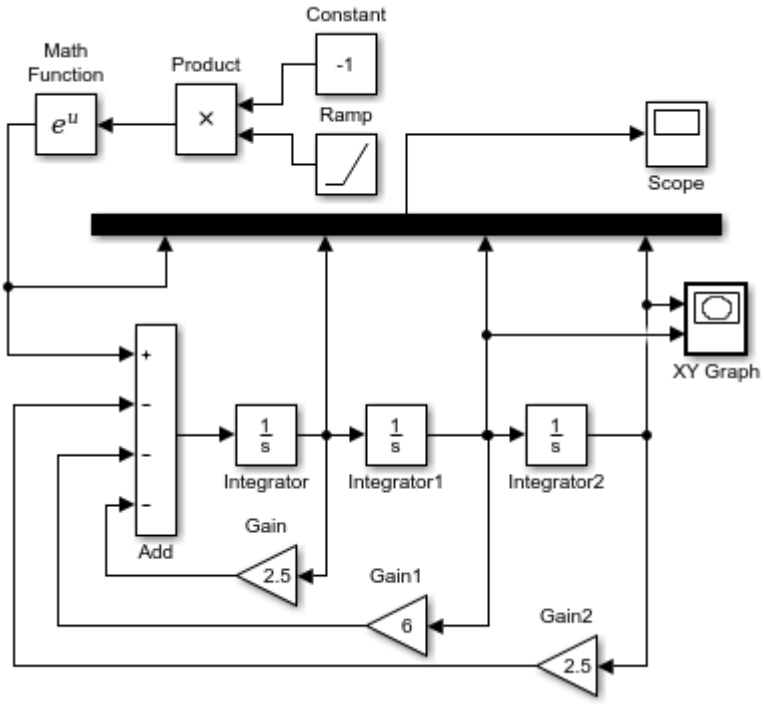


Рисунок 11.4. Структурная схема модели динамической системы 3 порядка

Коэффициенты уравнения устанавливаются в окнах параметров масштабных блоков Gain – Gain2. Начальные условия для функций и производных – в окнах параметров интеграторов Integrator – Integrator2. На рисунке 11.5 представлено окно параметров для интегратора Integrator, соответствующего $y''(0)=2$, начальные условия для интеграторов Integrator1 – Integrator2, соответствующие $y'(0)=-1$ и $y(0)=1$ соответственно задаются аналогично. Правая часть дифференциального уравнения сформирована с помощью блоков Ramp (генератор аргумента – t) и блока Math Function, настроенного на реализацию экспоненциальной функции. Визуализация переходного процесса показана на экране виртуального осциллографа (рисунок 11.6).

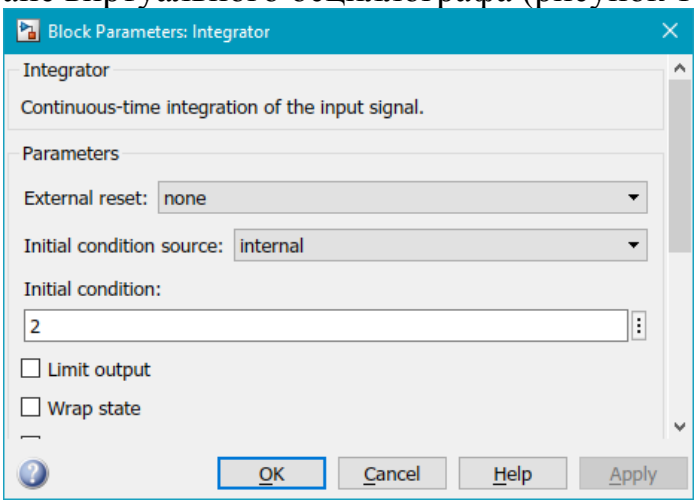


Рисунок 11.5 – Окно параметров функционального блока Integrator

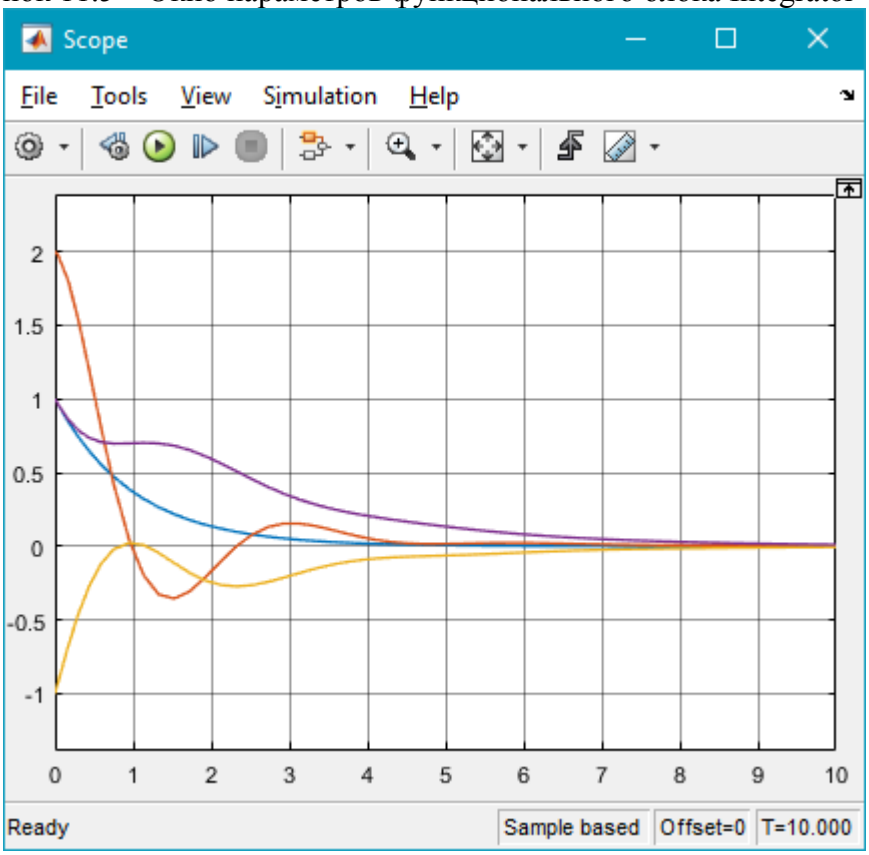


Рисунок 11.6 – Переходный процесс системы

Фазовый портрет системы на экране виртуального двух координатного регистратора (XY Graph) приведен на рисунок 11.7.

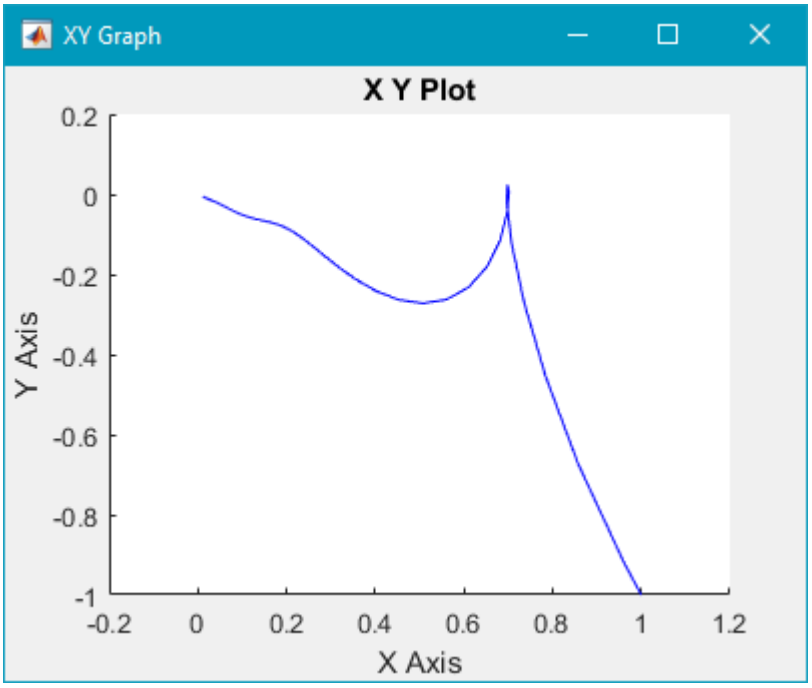


Рисунок 11.7. Фазовый портрет системы

Пример 3. Модель траектории полета тела, брошенного с начальной скоростью под углом к горизонту. Предположим, что наблюдатель, находясь над уровнем земли на высоте 1м, бросил камень под углом 30 градусов к горизонту с начальной скоростью 20 м/сек. Необходимо реализовать модель траектории полета камня под действием силы тяжести и определить расстояние от наблюдателя до точки падения камня. Влиянием атмосферы на полет камня пренебречь. Уравнения движения камня имеют вид:

$$y = y_0 + v \cdot \sin(\alpha) \cdot t - g \cdot \frac{t^2}{2}, \qquad x = v \cdot \cos(\alpha) \cdot t, \tag{11.4}$$

где: $y_0 = 1\,м, v = 20\,м/сек, \alpha = 30°, g = 9{,}81\,м/сек^2$.

Структурная схема модели траектории камня приведена на рисунке 4.7.

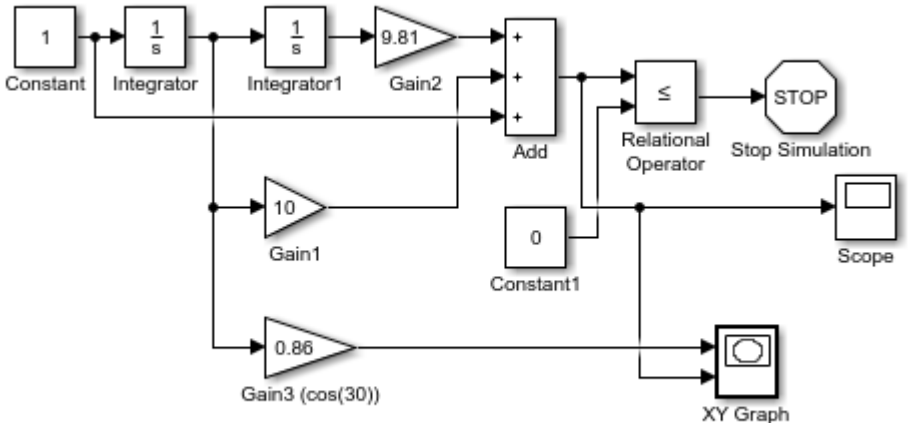


Рисунок 11.7. Структурная схема модели траектории камня

Вертикальная составляющая начальной скорости камня задается блоком Gain1, горизонтальная составляющая – блоком Gain3, ускорение земного тяготения – блоком Gain2. Значение текущей высоты полета камня как функции времени формируется на выходе сумматора. Сигнал окончания моделирования формируется блоками Relational Operator и Stop Simulation в момент времени, когда высота сравняется с нулем. Показания цифрового регистратора соответствует длине пути по горизонтали, пройденной камнем до момента соприкосновения с землей. На рисунках 4.8 и 4.9показаны соответственно траектория камня на экране виртуального двух координатного регистратора и график изменения во времени высоты полета камня.

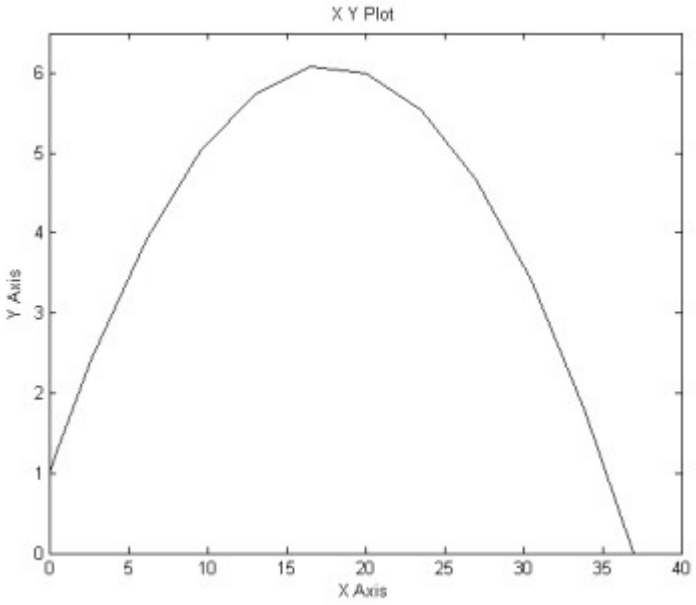


Рисунок 11.8. Траектория полета камня, брошенного под углом 30 градусов к горизонту с начальной высоты 1м, с начальной скоростью 20 м/сек

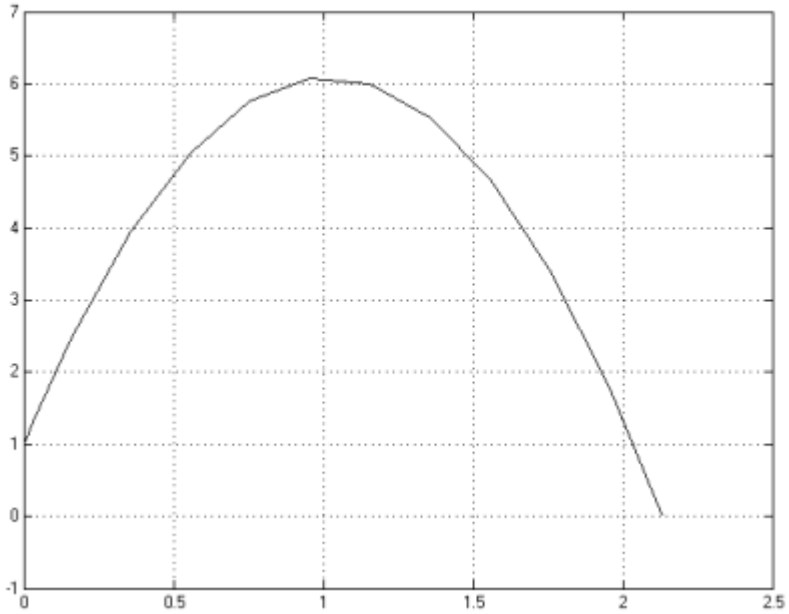


Рисунок 11.9. График изменения во времени высоты полета камня

Аппаратура и материалы. Для выполнения лабораторной работы необходимо использовать следующее: аппаратное обеспечение: персональный компьютер и программное обеспечение: операционная система Windows 10 и выше; Microsoft Office13 и выше, системы компьютерной математики Machcad и MATLAB R2017a и выше.

Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

Выполните предложенные задания, предварительно ознакомившись с теоретической частью.

Задание 11.1. Синтезировать структурную схему модели дифференциального уравнения 3 порядка:

$$\frac{d^2y}{dx^2} + A\frac{dy}{dx} + By = 0 \qquad (11.5)$$

С начальными условиями: $y(0), y'(0), y''(0)$. Построить фазовый портрет системы. Параметры сигналов приведены в таблице 4.1. Номер варианта соответствует номеру, под которым студент записан в списке группы.

Задание 11.2. Синтезировать структурную схему модели дифференциального уравнения 3 порядка:

$$\frac{d^3y}{dt^3} + C\frac{d^2y}{dx^2} + D\frac{dy}{dx} + Ey = 0 \qquad (11.6)$$

С начальными условиями: $y(0), y'(0), y''(0)$. Построить фазовый портрет системы. Параметры сигналов приведены в таблице 4.1. Номер варианта соответствует номеру, под которым студент записан в списке группы.

Задание 11.3. Построить модель траектории полета пушечного ядра под действием силы тяжести и определить расстояние от пушки до точки падения ядра, выпущенного с начальной скоростью V_0 под углом α к горизонту. Высота от уровня земли до дульного утолщения равна h. Влиянием атмосферы на полет ядра пренебречь.

Параметры задачи приведены в таблице 11.2. Номер варианта соответствует номеру, под которым студент записан в списке группы.

Таблица 11.2 – Параметры задач 1 и 2

№ вар-та	A	B	C	D	E	$y(0)$	$y'(0)$	$y''(0)$
1.	0.2	0.4	0.5	2	0.3	1	2	0
2.	1	0.8	3.2	1.4	0.6	-1	1	2
3.	3	4	4.2	1	0.4	0	2	1
4.	4.2	3	1	2	0.5	1	-1	-1
5.	4.2	1	0.4	4	4.2	2	1	0
6.	1	2	0.5	3	1	-1	0	1
7.	4	4.2	1	0.8	3.2	1	2	0
8.	3	1	2	4	4.2	-1	1	2
9.	1	0.4	4	3	1	-2	2	1
10.	2	0.5	3	1	0.4	1	0	1

Таблица 11.2. Параметры задачи 3

№ вар-та	V_0, *м/с*	α, *градусов*	h, *метров*
1.	32	30	1
2.	38	35	1,2
3.	34	32	2,2
4.	42	36	1,4
5.	37	34	0,4
6.	29	33	0,5
7.	52	30	2,8
8.	44	36	2
9.	47	33	4
10.	49	31	3

Содержание отчета и его форма

Подготовьте отчет, в котором приведите технологию выполнения заданий.

Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) формулировку задания и технологию его выполнения;
4) ответы на контрольные вопросы;
5) приложение – файлы выполненных заданий.

Вопросы для защиты работы

1. Приведите как минимум 3 примера динамических систем?
2. Какие основные математические блоки используются для моделирования динамических систем?
3. Дифференциальное уравнение какого порядка потребуется для описания модели физического маятника? Приведите общий вид такого уравнения.

ЛАБОРАТОРНАЯ РАБОТА 12
Разработка систем компьютерного моделирования динамических процессов в жидких дисперсных магнитных наносистемах

Цель и содержание: изучить основные приемы разработки систем компьютерного моделирования динамических процессов в пакете Simulink.

Организационная форма занятий: решение проблемных задач, разбор конкретных ситуаций

Вопросы для обсуждения на лабораторном занятии: построение х модели систем компьютерного моделирования динамических процессов в жидких дисперсных магнитных наносистемах в пакете Simulink системы MATLAB.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Для разработки системы компьютерного моделирования динамических процессов в жидких дисперсных магнитных наносистемах использована система дифференциальных уравнений ,описывающая процесс восстановления равновесной формы капли магнитной жидкости после выключения внешнего магнитного поля.

$$\left\{\begin{array}{l}\dfrac{dV_\lambda(t)}{dt}=-\dfrac{2}{3}\dfrac{\lambda(t)^2+2}{\lambda(t)\cdot(2\lambda(t)^2+1)}V_\lambda(t)^2-\dfrac{60\cdot\eta}{\left(\rho+\frac{\rho_1}{2}\right)\cdot R^2}\dfrac{\lambda(t)^{2/3}}{2\lambda(t)^2+1}\cdot V_\lambda(t)+\\ \quad -\dfrac{45\cdot\sigma}{4R^3\left(\rho+\frac{\rho_1}{2}\right)}\dfrac{\lambda(t)^3}{(2\lambda(t)^2+1)\cdot(\lambda(t)^2-1)}\left(\dfrac{\lambda(t)^2-4}{\sqrt{\lambda(t)^2-1}}\arcsin\left(\dfrac{\sqrt{\lambda(t)^2-1}}{\lambda(t)}\right)+\dfrac{2}{\lambda(t)^2}+1\right),\\ \dfrac{d\lambda(t)}{dt}=V_\lambda(t)\end{array}\right. \quad (12.1)$$

Состояние динамической системы характеризуется парой величин: координатой $\lambda(t)$ и скоростью $\dot{\lambda}(t)=V_\lambda(t)$, являющихся компонентами вектора состояний системы, принадлежащего пространству состояний (иначе, фазовому пространству).

Для решения системы (12.1) необходимо составить блок схему, позволяющую исследовать поведение изучаемой динамической системы в пакете расширения Simulink системы Matlab. В основе построения блок схемы лежит последовательный итерационный процесс. Алгоритм итерационного процесса заключается в следующем:

1. Формируется правая часть первого уравнения системы (12.1), которая зависит от относительного удлинения капли $\lambda(t)$ и скорости изменения удлинения $V_\lambda(t)$. Считается, что $\lambda(t)$ и скорость изменения

$\lambda(t)$ – $V_\lambda(t)=\dot{\lambda}(t)$ известны, следовательно, известно и ускорение $a_\lambda(t)=\ddot{\lambda}(t)$.

2. Определяется скорость $V_\lambda(t)=\dot{\lambda}(t)$ путем интегрирования ускорения $a_\lambda(t)$.

3. Определяется относительное удлинение капли $\lambda(t)$ путем последующего интегрирования скорости $V_\lambda(t)$.

4. Выполняется п.1 – полученные значения $\lambda(t)$ и $V_\lambda(t)$ используются для формирования правой части первого уравнения системы (12.1)

Блок схема (рисунок 21.1) содержит два последовательно соединенных интегратора с внешне задаваемыми величинами ρ, ρ_1, σ, η, R в соответствии с системой (21.1) и начальными условиями $\lambda(0)=\lambda_0$ и $V_\lambda(0)=\dot{\lambda}_0$.

На вход первого интегратора подается ускорение $\dot{V}_\lambda(t)$, в качестве начального условия используется начальное значение скорости $V_\lambda(0)=\dot{\lambda}_0$.

На выходе этого интегратора будет текущая скорость изменения относительного удлинения капли $V_\lambda(t)$. Эта величина подается на вход второго интегратора с начальным условием в виде начального удлинения $\lambda(0)=\lambda_0$.

Выходом со второго интегратора будет зависимость относительного удлинения капли от времени.

Отдельным блоком в схеме приведена подсистема – субмодель (блок Sybsystem), представляющая собой функцию правой части первого уравнения системы (12.1) – рисунок 12.2. При построении блок схемы подсистемы ее связь с основной системой осуществляется путем ввода в подсистему стандартных блоков In (Вход) и Out (Выход). Все величины, которые формируются в основной модели системы, а используются в подсистеме (субмодели) поступают в подсистему через блоки In.

Величины, сформированные в подсистеме, а используемые в основной системе, поступают в основную модель из подсистемы через блоки Out. При этом на изображении блока подсистемы (блок Sybsystem) в блок-схеме основной модели появляется количество входов, равное числу введенных в подсистему блоков In (рисунок12.1).

Входными параметрами подсистемы являются $\lambda(t)$ и $\ddot{\lambda}(t)$, а входными величинами ρ, ρ_1, σ, η, R.

Для отображения результатов моделирования использован блок Scope, подключенный к сформированному сигналу. Для отображения фазового портрета использован блок XYGraph.

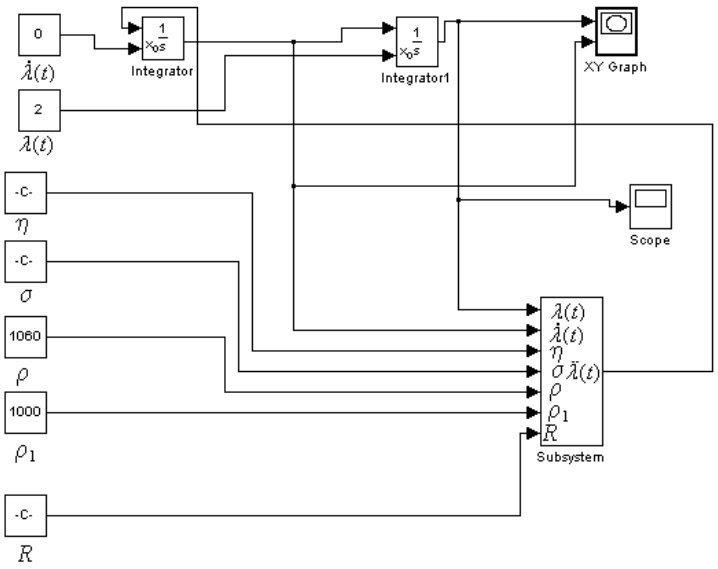


Рисунок 12.1 – Блок-схема основной модели динамической системы, описываемой системой уравнений (12.1)

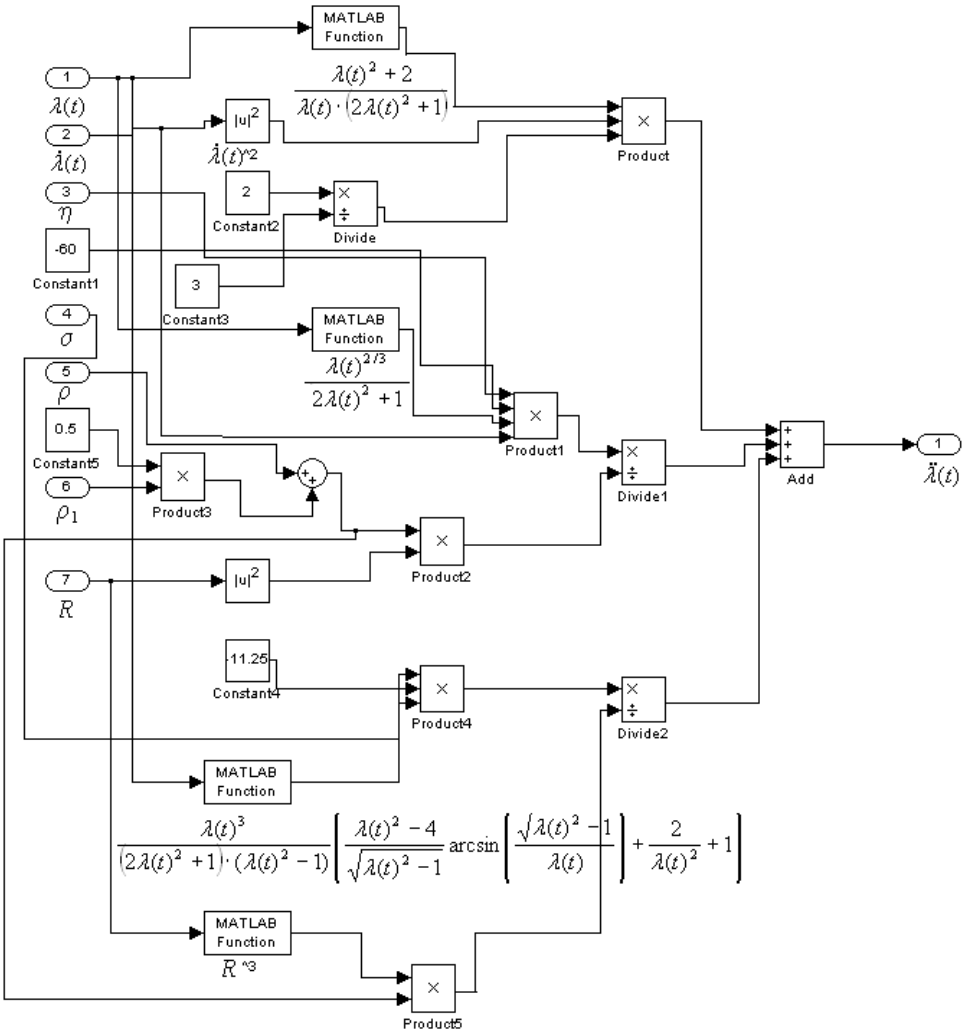


Рисунок 12.2 – Блок-схема субмодели динамической системы (12.1)

Аппаратура и материалы. Для выполнения лабораторной работы необходимо использовать следующее: аппаратное обеспечение:

персональный компьютер и программное обеспечение: операционную систему Windows 10 и выше; Microsoft Office13 и выше, системы компьютерной математики Machcad и MATLAB R2017a и выше.

Указание по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения; ремонт персонального компьютера. Соблюдать правила технической эксплуатации и техники безопасности при работе с электрооборудованием.

Методика и порядок выполнения работы

Выполните предложенные задания, предварительно изучив материал, представленный в теоретических указаниях.

Задание 12.1. Разработайте структурную схему модели, описывающей динамические процессы – рисунок 12.1

Задание 12.2. Разработайте структурную схему субмодели, описывающей динамические процессы – рисунок 12.2

Задание.12.3. Выполните моделирование, используя следующие параметры:R = 3 мм, ρ = 1060 кг/м^3, ρ_1 = 1000 кг/м^3,σ = 2·10^{-3} Н/м, $\eta = 0{,}0043$ Па · с.

Результаты моделирования представлены на рисунке 12.3, 12.4.

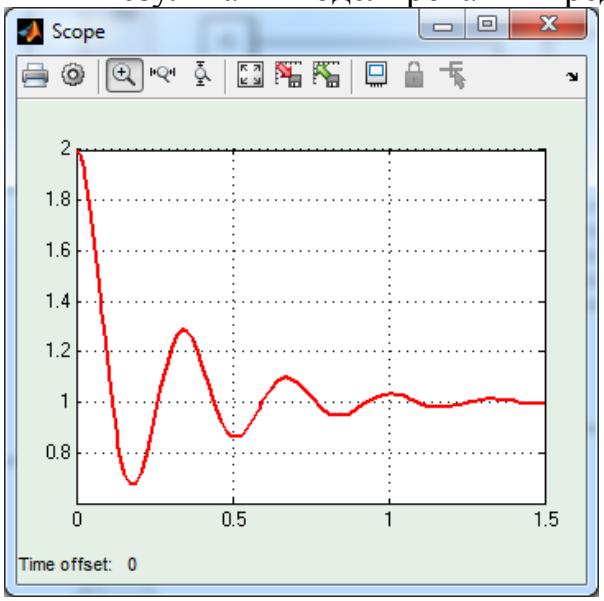


Рисунок 12.3 – Результат решения системы уравнений (12.1) в пакете расширения Simulink системы Matlab

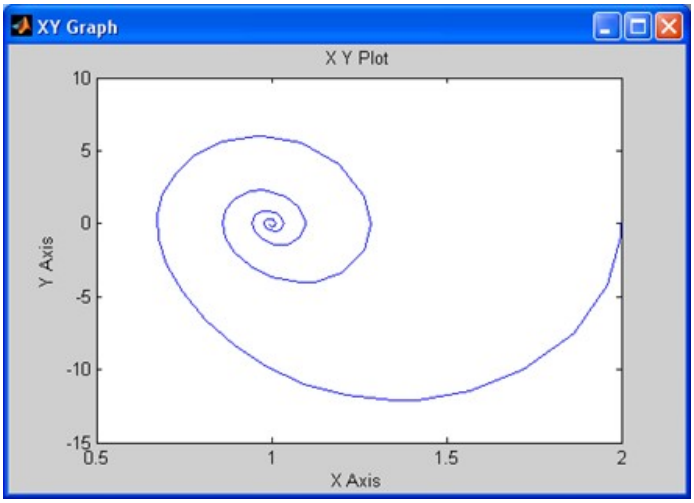


Рисунок 12.4 – Фазовый портрет, полученный в результате решения системы уравнений (12.1) в пакете расширения Simulink системы Matlab

Фазовый портрет, полученный с помощью блока XYGraph пакета Simulink, представлен на рисунке 12.4. Точка с координатами (1,0) на рисунке является асимптотически устойчивым фокусом. Таким образом, эта точка является аттрактором нулевой размерности. Наличие на фазовой траектории фокуса свидетельствует о том, что рассматриваемая динамическая система является диссипативной.

Задание 12.4. Индивидуальное задание (таблица 12.1)

Вариант	R, мм	ρ ,кг/м^3	ρ_1,кг/м^3	$\sigma \cdot 10^{-3}$, Н/м	η .Па $\cdot$ с
1	3	1060	1000	2	0,0033
2	5	1100	1160	3	0,0043
3	4	1020	1160	5	0,0053
4	7	1080	1030	1	0,0023
5	8	1040	1005	8	0,093
6	2	1050	1040	10	0,0073
7	10	1090	1060	4	0,0013
8	6	1010	1000	2	0,0063
9	9	1070	1020	6	0,0083
10	15	1030	1000	7	0,0039

Содержание отчета и его форма

Подготовьте отчет, в котором приведите технологию выполнения заданий.

Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) формулировку задания и технологию его выполнения;
4) ответы на контрольные вопросы;
5) приложение – файлы выполненных заданий.

Вопросы для защиты работы

1. Как осуществляется разработка структурных схем?

2. Как с помощью Simulink можно моделировать поведениесложных систем?

3. На какой технологии основана разработка моделей средствами SIMULINK (S-модели)?

ЛАБОРАТОРНАЯ РАБОТА 13
Моделирование информационных процессов в среде ARENA

Цель и содержание: изучить основные приемы работы со средой имитационного моделирования ARENA и приобрести навыки построения дискретно-событийных моделей.

Организационная форма занятий: решение проблемных задач, разбор конкретных ситуаций

Вопросы для обсуждения на лабораторном занятии: технология построения модели «Магистраль передачи данных» средствами среды имитационного моделирования ARENA.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Особым видом математических моделей являются имитационные модели. Имитационная модель – это компьютерная программа, которая описывает структуру и воспроизводит поведение реальной системы во времени. Можно сказать, что имитационная модель – это набор правил, согласно которым система переходит из одного состояния в другое.

Правила могут задаваться самыми различными способами, например, дифференциальными уравнениями, диаграммами состояний, диаграммами процессов, расписаниями. Выходные данные модели всегда можно проанализировать прямо по ходу моделирования.

Имитационные модели разрабатываются с помощью специализированного программного обеспечения, в котором используются различные языки моделирования. Имитационное моделирование – разработка компьютерных моделей и постановка экспериментов на них. Можно выделить основные преимущества имитационного моделирования по сравнению с аналитическим.

1. Имитационные модели позволяют анализировать системы и находить решения в тех случаях, когда такие методы, как аналитические вычисления не справляются с задачей.
2. Разрабатывать имитационную модель гораздо проще, чем аналитическую, поскольку процесс создания модели будет инкрементальным (пошаговый, усложняющийся постепенно) и модульным.
3. Структура имитационной модели естественным образом отображает структуру моделируемой системы.
4. Имитационная модель позволяет отслеживать все объекты системы, учтенные в выбранном уровне абстракции, добавлять метрики и проводить статистический анализ.
5. Имитационная модель дает возможность проигрывать модель во времени и анимировать ее поведение.

Анимация будет неоспоримым преимуществом при демонстрации модели и может оказаться полезной для верификации (тестирование на предмет соответствия проекта техническому заданию –замыслу проектировщика).

В имитационном моделировании под методом понимается некая основа, которую мы используем, чтобы «перевести» систему из реального мира в мир моделей. Метод предполагает определенный язык, положения и условия для разработки модели. На данный момент, существует три метода (концепции) моделирования:

- системная динамика и динамические системы;
- дискретно-событийное моделирование;
- агентное моделирование.

Каждый метод применяется в некотором диапазоне уровней абстракции. Системная динамика предполагает очень высокий уровень абстракции и, как правило, используется для стратегического моделирования. Дискретно-событийное моделирование поддерживает средний и низкий уровни абстракции. Между ними находятся агентные модели, которые могут быть как очень детализированными, когда агенты представляют физические объекты, так и предельно абстрактными, когда с помощью агентов моделируются конкурирующие компании или правительства государств.

Пример 1. Постановка задачи «Магистраль передачи данных»

Магистраль передачи данных состоит из двух каналов (основного и резервного) и общего накопителя. При нормальной работе сообщения передаются по основному каналу за 7 ± 3 мкс. В основном канале происходят сбои через интервалы времени 200 ± 35 мкс. Если сбой происходит во время передачи, то за 2 мкс запускается запасной канал, который передает прерванное сообщение с самого начала в течение 8 ± 3 мкс. Восстановление основного канала занимает 23 ± 7 мкс. После восстановления резервный канал выключается и основной канал продолжает работу с очередного сообщения. Сообщения поступают через 9 ± 4 мкс. и остаются в накопителе до окончания передачи. Если в накопителе в очереди оказывается более 10 сообщений, то очередное сообщение оказывается потерянным. В случае сбоя передаваемое сообщение передается повторно по запасному каналу. Смоделировать работу магистрали передачи данных в течение 1 мкч. Определить загрузку основного и запасного каналов, частоту отказов канала и число потерянных сообщений.

Разработка модели магистрали в среде ARENA Для выполнения практических заданий по моделированию в системе ARENA нужно скачать свободно распространяемую Free Trial версию (время ее использования не ограничено) с официального сайта разработчика http://www.arenasimulation.com и установить ее компьютере.

Интерфейс среды имитационного моделирования ARENA На рисунке 13.1 показан интерфейс среды имитационного моделирования ARENA.

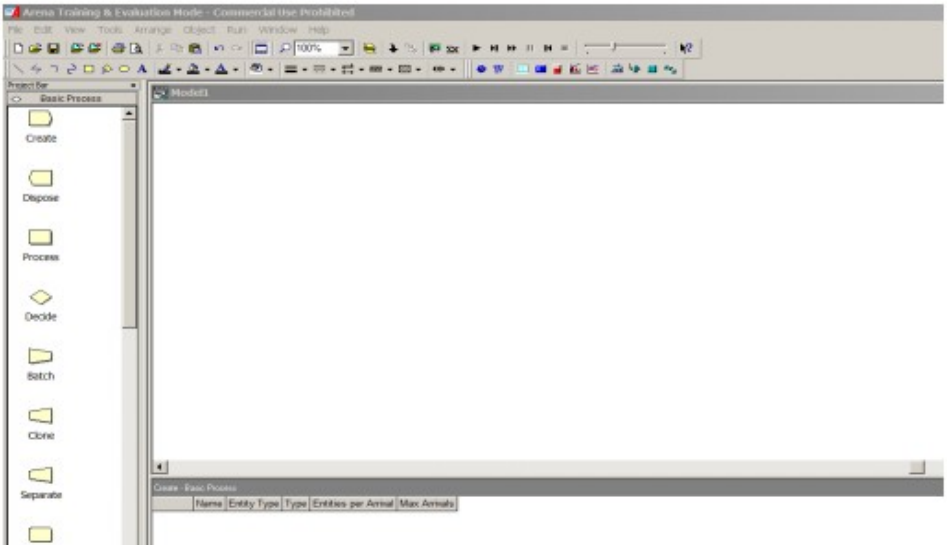


Рисунок 13.1 – Интерфейс среды имитационного моделирования ARENA

Сверху располагается панель инструментов, по центру – рабочая область (область диаграммы, графическая область), внизу – окно свойств модулей, слева – окно проекта. В Окне рабочего поля модели описывается логика модели с использованием схемных (графических) модулей. Окно рабочего поля представляет графику модели, включая блок-схему процесса, анимацию и другие элементы. Окно свойств модулей отображает свойства всех модулей (как модулей данных, так и схемных), имеющихся и используемых в модели. Окно проекта (слева) – это навигатор системы, в котором отображается рабочая панель со всеми модулями и другие доступные и открытые панели. Окно проекта включает в себя несколько панелей:

1. Basic Process Panel (панель основных процессов) – содержит модули схемные (графические) и невизуальные, которые используются для моделирования основной логики системы, в них отображаются свойства различных объектов (транзактов, ресурсов, очередей, атрибутов, переменных, расписаний и т.д.).

2. Advanced Process Panel (панель усовершенствованных процессов) – содержит дополнительные модуле для создания моделей со сложной логикой процесса.

3. Advanced Transfer Panel (панель перемещения) – содержит специально разработанные блоки для моделирования процесса перемещения объектов с помощью транспортера или конвейера.

4. Reports (панель отчетов) – панель сообщений: содержит сообщения, которые отображают результаты имитационного моделирования.

5. Navigate (панель навигации) – панель управления позволяет отображать все виды модели, включая управление через иерархические подмодели. Кроме этих основных панелей, могут быть и другие.

Схема модели. На рисунке 13.2 изображена уже готовая схема рассматриваемой модели. Для удобства в рабочее окно помещена постановка задачи в виде текстовой области. Все схемные модули вводятся в рабочее поле перетаскиванием из окна проекта. Модули соединяются автоматически или с помощью коннектора на панели инструментов. Свойства схемных модулей задаются в окнах параметров, которые появляются при двойном щелчке по изображению модуля. Невизуальные модули вызываются щелчком мыши.

При разработке простых моделей невизуальные модули заполняются автоматически, когда создается схема, и в окнах параметров вводятся свойства объектов. Разрабатываемый проект сохраняется в выбранном каталоге как файл с расширением Arena Model Document (.doe). Заметим, что инструменты реализации модели могут быть различными. Здесь предлагается один из вариантов, может быть, не самый рациональный. В графе модели, представленной на рисунке 13.2, используются две несвязанные графически ветви (два подграфа), которые функционируют параллельно.

Рисунок 13.2 – Схема модели имитационного моделирования работы магистрали передачи данных

В создаваемой модели введены два транзакта: первым транзактом (сущностью) является поток данных EntityMessage (сообщений), второй транзакт (фиктивный) Trouble – сбой в работе основного канала. Транзакты поступают в систему через модули создания транзактов. Они обозначены в модели CreateMessage, CreateTrouble.

Опишем каждую из ветвей моделирования. Описание моделирования сбоя в канале связи В верхней ветви (рисунок 13.3)

модели (рисунок 13.2) фиктивные транзакты Trouble вводятся модулем CreateTrouble. Модуль помещается в рабочее окно перетаскиванием из панели Basic Process в окне проектов.

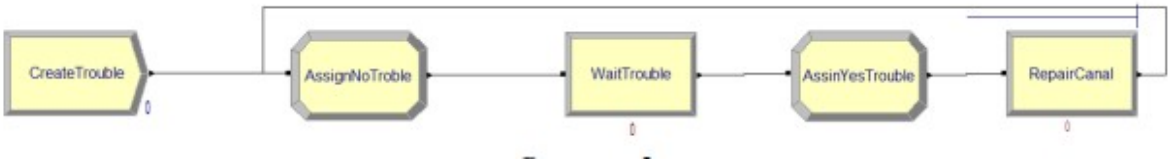


Рисунок 13.3 – Схема моделирования сбоя в канале связи

В окне параметров (рисунок 13.4) этого модуля (вызывается двойным щелчком по модулю) задаем новые имя модуля и тип (транзакта) сущности. В принципе, можно было оставить стандартные имена и типы, но для понимания логики работы модели их лучше поменять. Создадим такую реализацию, когда транзакт-сбой будет только один, но он будет с определенной в задаче периодичностью появляться в системе. Исходя из этого, заполним поля свойств, как показано на рисунке 13.4.

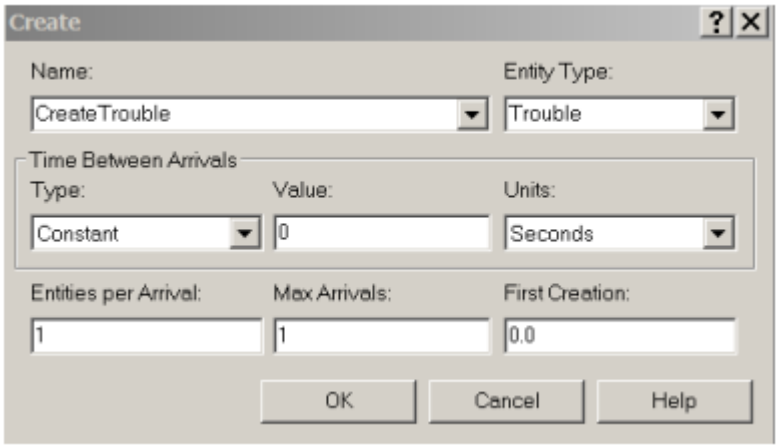


Рисунок 13.4 – Окно параметров модуля CreateTrouble

Модельное время будем условно измерять в секундах, время первого сбоя не известно, в этом случае по умолчанию в поле First Creation записывается 0.

В окне параметров модуля AssignNoTroble типа Assign на панели Basic Process (рисунок 13.5) стандартное имя модуля заменим на имя AssignNoTroble. Кроме того, присвоим значение логической переменной, которая будет фиксировать наличие-отсутствие сбоя в основном канале, strouble=0, что означает, что изначально сбоя в канале нет.

В следующем блоке (рисунок 13.6) WaitTrouble типа Process (его имя изменено по логике задачи) реализовано ожидание сбоя основного канала, который случается с определенной в задаче периодичностью.

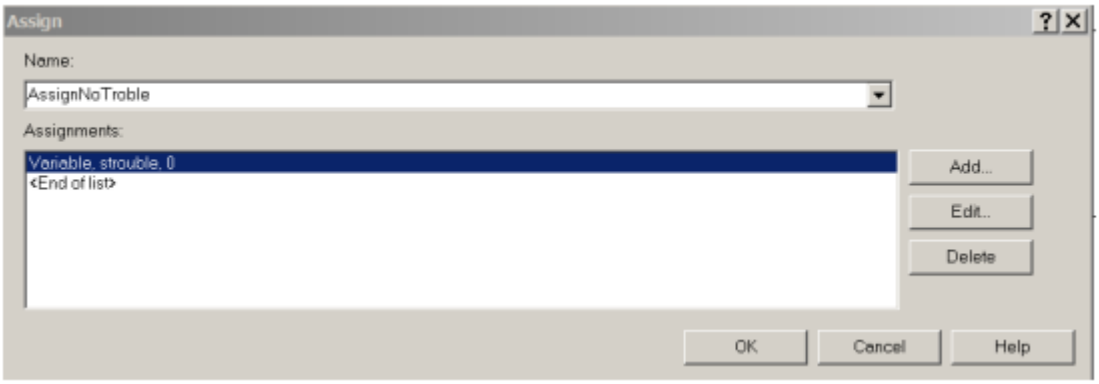


Рисунок 13.5 – Окно параметров модуля AssignNoTroble

Так как в период ожидания не задействованы ресурсы, используемые в задаче (о ресурсах речь пойдет ниже), то используется действие Delay, что означает задержку. Выбран треугольный закон распределения времени задержки. Данные в поля внесены в соответствии с условием задачи.

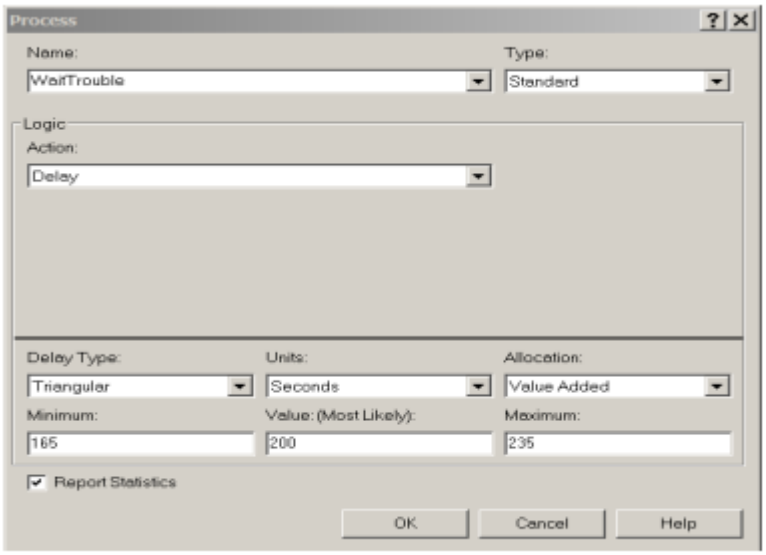


Рисунок 13.6 – Окно диалога Process

В модуле AssignYesTroble типа Assign (рисунок 13.7) присваивается значение логической переменной, которая фиксирует наличие сбоя в основном канале, strouble=1, что означает, что сбой наступил. Кроме того, предыдущее значение переменной ntrouble увеличивается на единицу. Эта переменная служит для подсчета числа сбоев в канале. Средствами визуализации ее значения отображаются в рабочей области. Как это делается, будет описано ниже в разделе анимация.

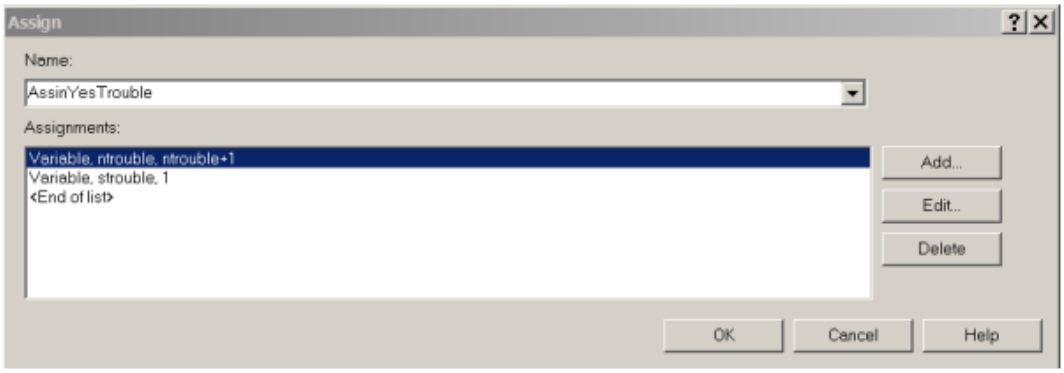


Рисунок 13.7 – Окно диалога Assign

Завершающим блоком ветви является модуль Process (рисунок 13.8), которому дано имя RepairCanal. Этот модуль реализует процесс ремонта основного 11 канала. Так как при этом ресурс CanalMain, соответствующий основному каналу, будет задействован и не может использоваться для передачи сообщений, то в окне свойств этого модуля нужно выбирать действие SeizeDelayRelease (захват-задержка-освобождение ресурса). Выбираем треугольный закон распределения времени задержки. Данные вводим, как определено в условии задачи.

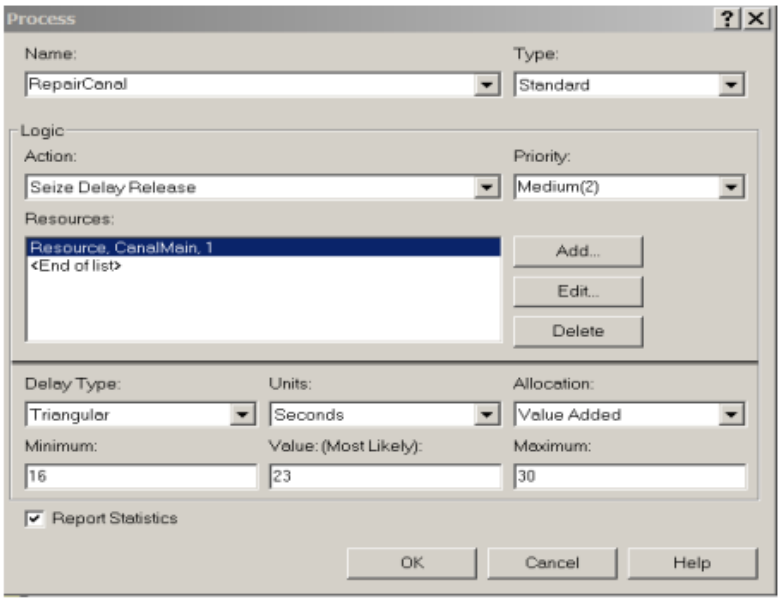


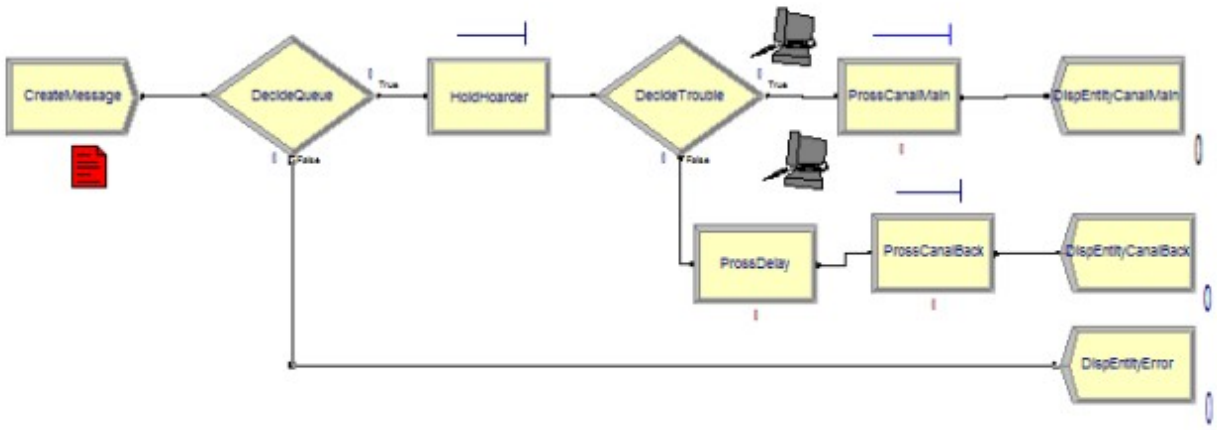
Рисунок 13.8 – Окно диалога модуля Process для выбора параметров

Вся схема должна заканчиваться либо модулем выхода транзактов типа Dispose, либо быть циклической. В нашем случае для реализации описанной логики, согласно которой один транзакт-сбой циклически функционирует в системе, нужно добавить коннектор, соединяющий выход блока RepairCanal с входом блока AssignNoTroble, как показано на рисунке 13.3.

Пример 2 Моделирование процесса передачи сообщений.

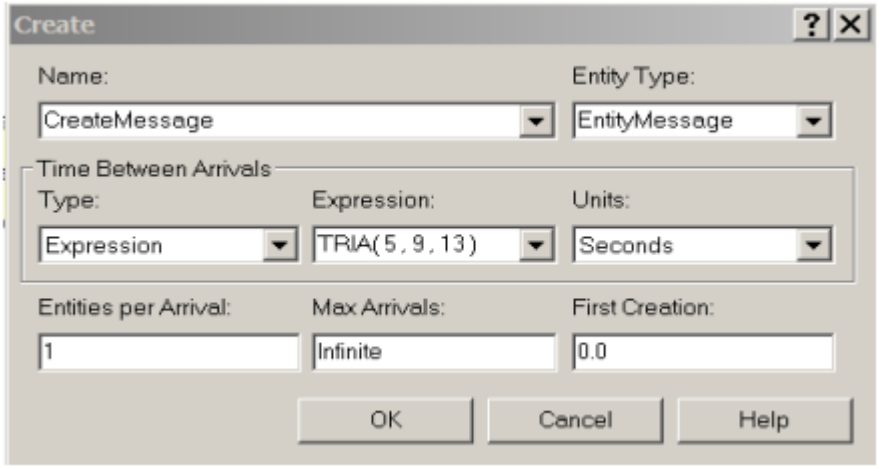
На рисунке 13.9 изображена вторая (нижняя) ветвь модели. В качестве ресурсов используются два канала передачи данных, которые задействованы в модулях типа Process, обозначенных как ProssCanalMain,

ProssCanalBack. Ввод сообщений в систему имитируется модулем «CreateMessage».



13.9 – Схема моделирования процесса передачи сообщений

В окне параметров (рисунок 13.10) этого модуля (вызывается двойным щелчком по модулю) задаем новые имя модуля и тип сущности.



13.10 – Окно диалога Create с параметрами закона распределения TRIA

Зададим треугольный закон распределения TRIA (5 , 9 , 13) времени между прибытием соседних сообщений. Модельное время будем условно измерять в секундах, сообщения прибывают по одному (Entities per Arrival), количество сообщений неограниченно (Infinite), первое сообщение прибывает с начала отсчета модельного времени «0». Модуль DecideQueue типа Decide организует разветвление потока сообщений. Если длина очереди в накопителе меньше 10, то сообщения попадают в систему и обслуживаются. В противном случае сообщения теряются, т.е. выходят из системы (блок DispEntityError). На рисунке 13.11 показано окно параметров этого модуля: выбран тип (два перехода по условиям, заданным выражением) 2-way by Condition. При вводе условий перехода используется построитель выражений Expression Builder, который вызывается из контекстного меню путем наведения курсора на поле Value и нажатия правой кнопки мыши.

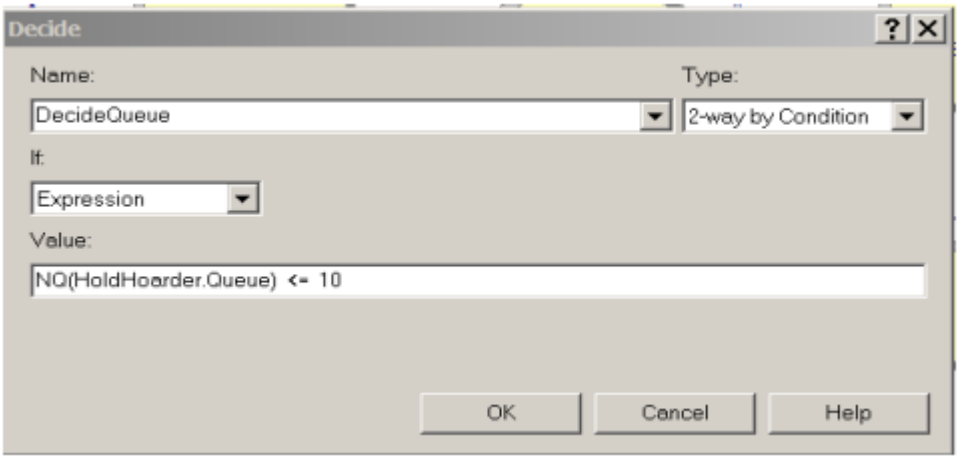


Рисунок 13.11 – Окно Decide модуля DecideQueue

При наборе выражения (рисунок 13.12) следует использовать кнопки окна построителя для ввода логических операций. Кроме того, используются встроенная переменная Current Number in Queue – число транзактов в очереди в данный момент из меню Basic Process Variable/Queue.

Поступление сообщений в накопитель и их задержка до освобождения каналов моделируется модулем HoldHoarder типа Hold (шлагбаум), в котором создается очередь. Этот модуль нужно взять с панели Advanced Process окна проектов. В окне параметров (рисунок 13.13) этого модуля опишем условия прохода сообщения через шлагбаум

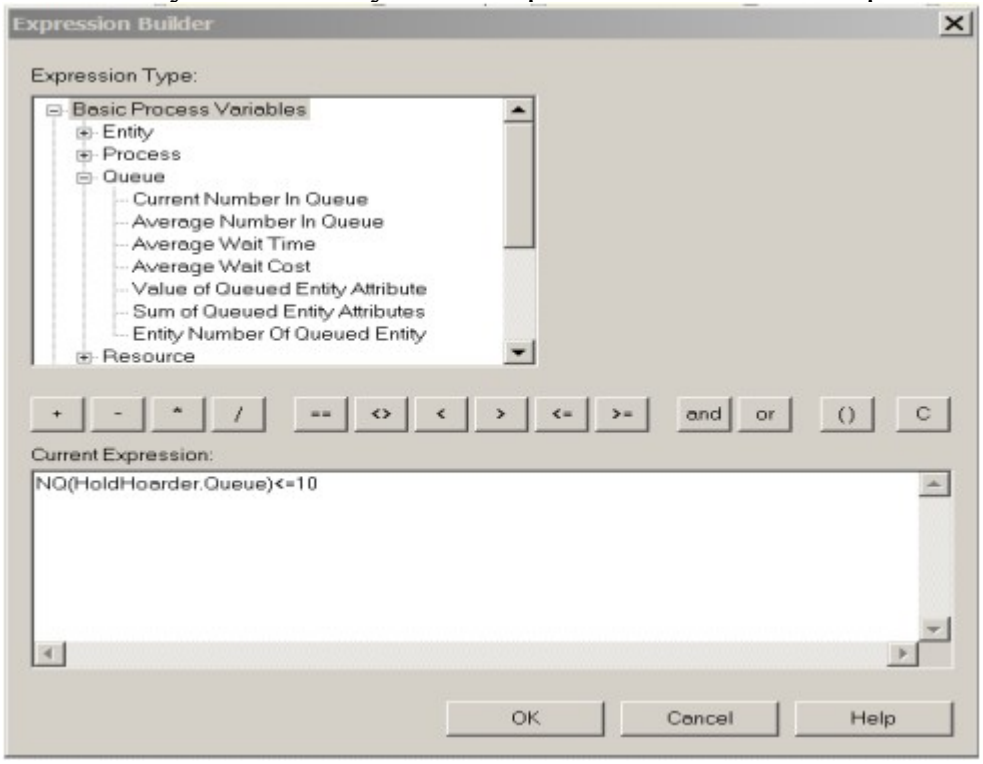


Рисунок 13.12 – Окно построителя выражений Expression Builder

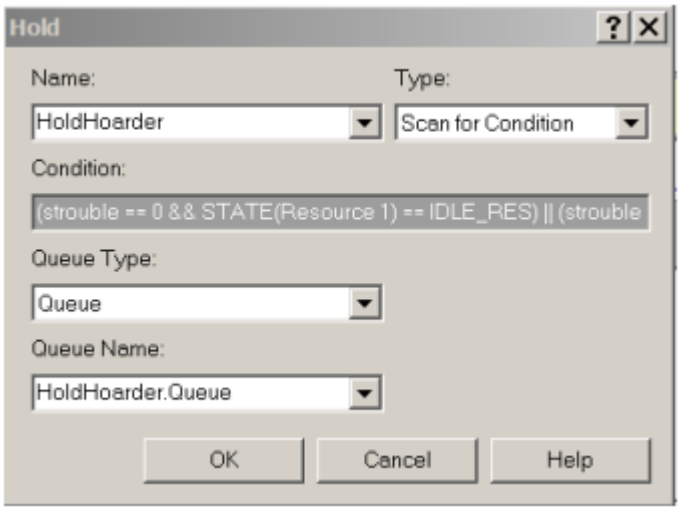


Рисунок 13.12 – Окно Hold модуля HoldHoarder

При вводе условий перехода используется построитель выражений Expression Builder, который вызывается из контекстного меню путем наведения курсора на поле Condition и нажатия правой кнопки мыши.

В поле Current Expression окна построителя составим выражение (рисунок 13.14), означает, что сообщения обрабатываются основным каналом, если в нем нет сбоя, и резервным каналом, если есть сбой в основном канале. При наборе выражения следует использовать кнопки окна построителя для ввода логических операций. Кроме того, используются встроенная переменная Current State и константа Idle State Constant состояния ресурса из меню Basic Process Variable/Resource/State.

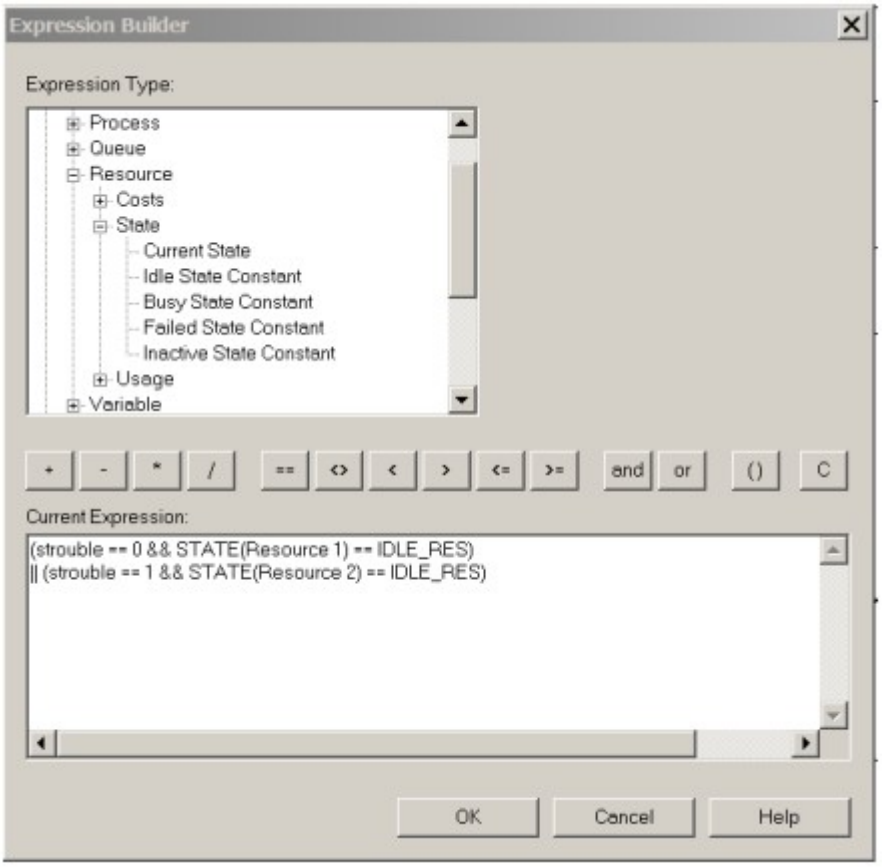


Рисунок 13.14 – Окно построителя выражений Expression Builder с заполненным полем Current Expression

Модуль DecideTrouble типа Decide организует разветвление потока сообщений. Если нет сбоя в основном канале (strouble == 0), то сообщения обслуживаются им, в противном случае – резервным. На рисунке 13.15 показано окно параметров этого модуля.

Модули процессов ProssCanalMain и ProssCanalBack выполняют передачу сообщений по основному каналу или резервному, когда основной в нерабочем состоянии, при этом задействуются ресурсы CanalMain и CanalBack, соответствующие этим каналам в количестве 1 канал. Модулю ProssCanalBack предшествует процесс задержки сообщения на 2 мкс. по условию задачи (модуль ProssDelay). На рисунке 13.16 показано окно параметров этого модуля.

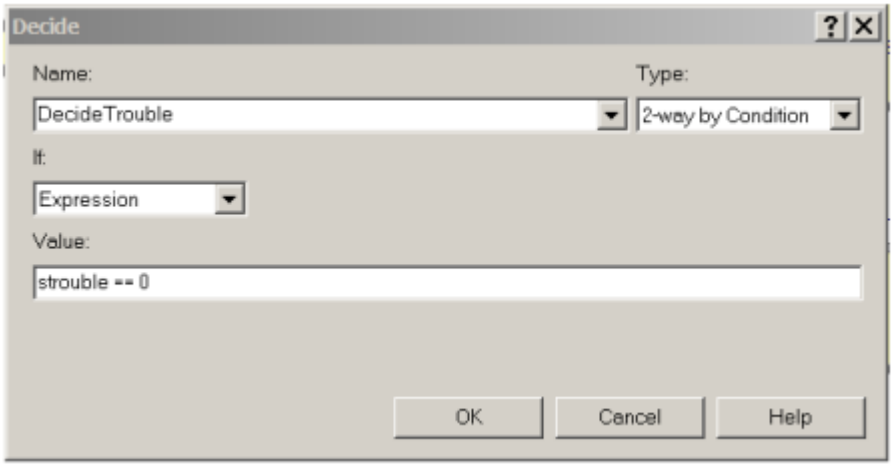


Рисунок 13.15 – Окно диалога Decide модуля DecideTrouble с заполненными полями

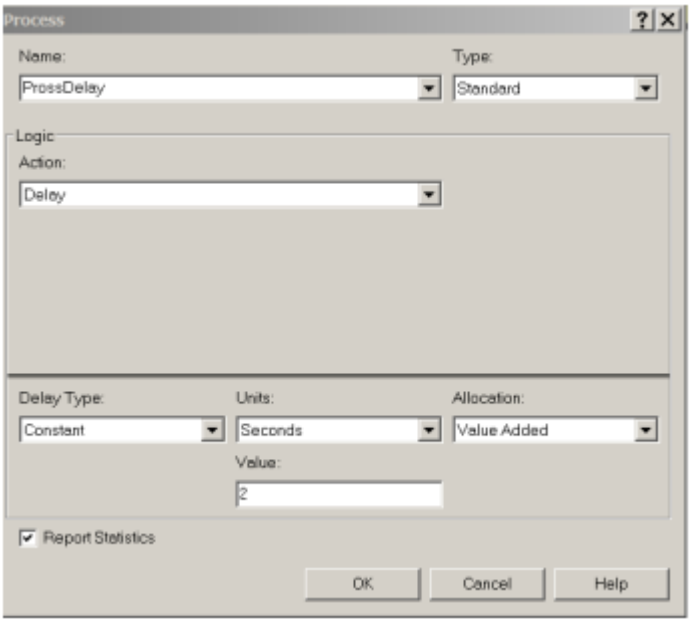


Рисунок 13.16 – Окно модуля ProssDelay

На рисунке 13.17 представлено окно параметров модуля ProssCanalBack. Для задания количества ресурсов (одна единица)

используется кнопка Add. 16 Пройдя через резервный канал, сообщения выходят из системы (модуль DispEntityCanalBack).

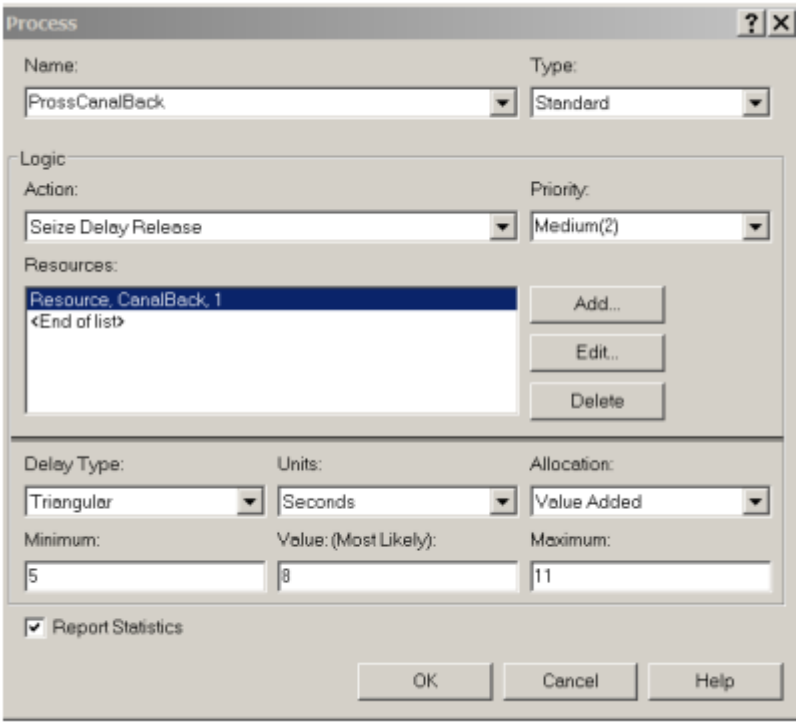


Рисунок 13.17 – Окно модуля ProssCanalMain

На рисунке 18 представлено окно параметров модуля ProssCanalMain. время прохождения одного сообщения распределено по треугольному закону с заданными по условию параметрами. Пройдя через основной канал, сообщения выходят из системы (модуль DispEntityCanalMain).

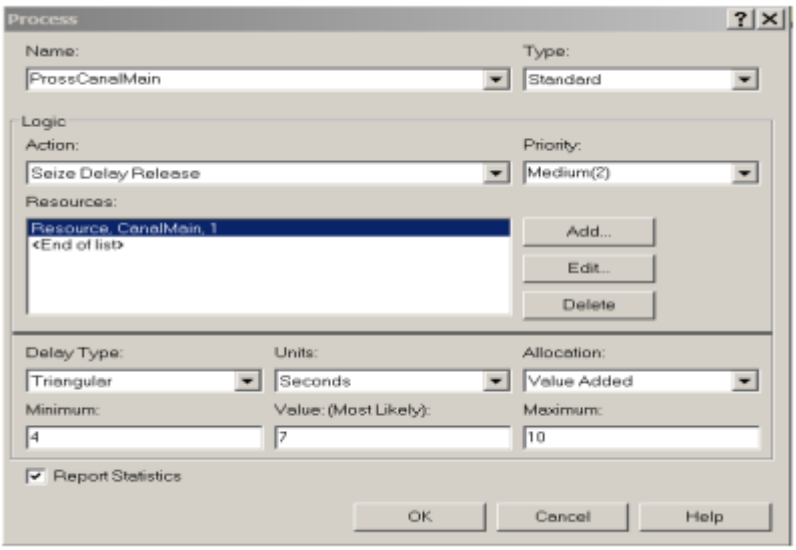


Рисунок 13.17 – Окно для ввода параметров модуля ProssCanalMain

Пример 3. Анимация в модели.

На рабочее поле рядом со схемой (рисунки 13.2, 13.19) вынесены элементы анимации:

- Cloke , здесь в ходе проигрывания модели отражается текущее время моделирования (Current Simulation Time – TNOW);

- текущие значения переменных (ntrouble, strouble);

- светофор , красный цвет светофора сигнализирует о наличии сбоя;

- графики зависимости числа сообщений, проходящих по основному каналу (красный, более крутой) и резервному (зеленый, более пологий), в зависимости от времени;

- Картинки, соответствующие транзактам и ресурсам.

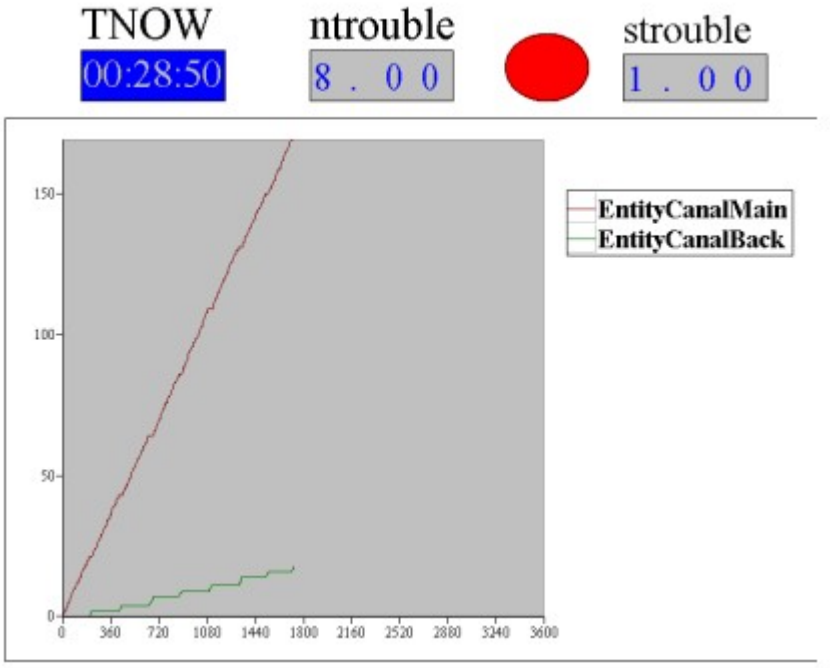


Рисунок 13.19 – Результаты анимации

Все элементы анимации взяты с панели инструментов. Чтобы вставить элемент анимации, нужно нажать на него левой кнопкой мыши. В появившемся окне параметров настроить свойства, затем нажать Ok. В результате элемент появится в рабочей области, с помощью мыши настроить его положение в рабочей области. Далее, зажав левую кнопку мыши, настроить его размер, затем щелчком левой кнопки зафиксировать элемент в рабочей области.

В окне параметров элемента анимации можно измерить его свойства, например, форма светофора изменена со стандартной-прямоугольной на круглую.

Для вызова окна свойств графика Plot нужно дважды щелкнуть по области графика. На вкладке Data Series для каждого графика в поле Name ввести 18 его имя (EntityCanalMain, EntityCanalMain), которое будет отображаться на экране. В поле Expression выбрать из раскрывающегося списка встроенную переменную (ProssCanalMain.NumberOut, ProssCanalBack.NumberOut), которая определяет количество сообщений, прошедших через основной или резервный канал в зависимости от

времени. Зависимости этих переменных от времени будут отображаться на графиках.

Для удобного отображения графиков можно скорректировать максимальные и минимальные значения отображаемых переменных, шаги по осям *Ox* и *Oy*: вкладка Axes, элемент scale, свойство MajorIncrem.. . В окне параметров Plot возможна настройка и других элементов графика. Можно также настроить дополнительные картинки для транзактов и ресурсов, нажимая на панели инструментов кнопки  и  .

Заметим, что стандартные картинки выбираются из раскрывающегося списка в невизуальном модуле свойств Entity на панели Basic Process (рисунок 13.20). Окно невизуального модуля вызывается двойным щелчком.

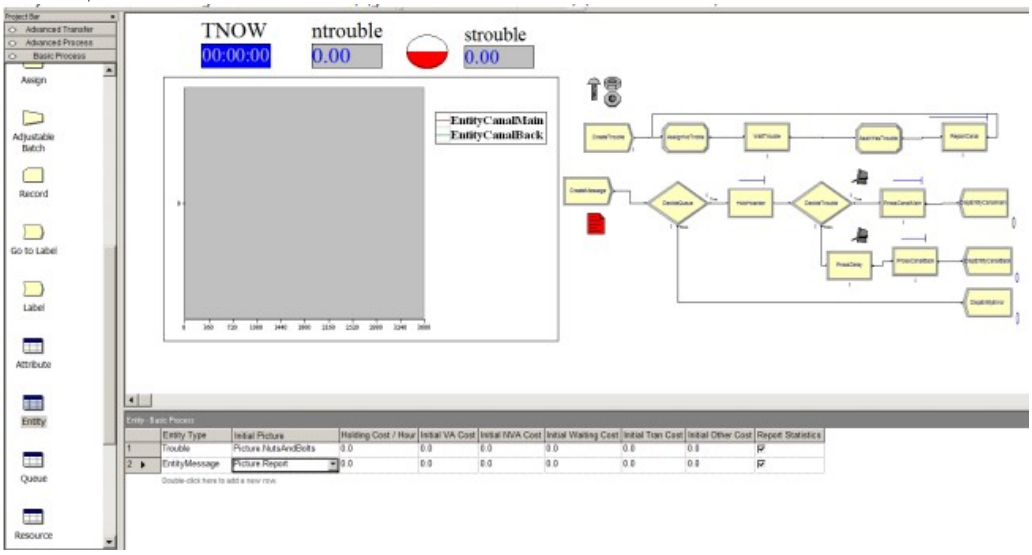
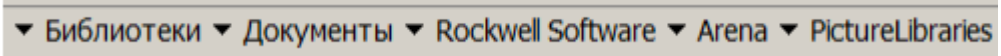


Рисунок 13.20 – Настройка свойств анимации с помощью модулясвойств Entity

При нажатии на кнопку  появляется окно настройки дополнительных картинок для Entity, представленное на рисунке 13.21. Справа отображаются картинки из открытой библиотеки. Чтобы открыть другую библиотеку, нужно нажать Open и выбрать библиотеку из раскрывающегося списка библиотек. Если список пуст, то нужно указать путь к библиотекам. При установке Arena по умолчанию библиотеки обычно размещаются в папке по следующему пути:



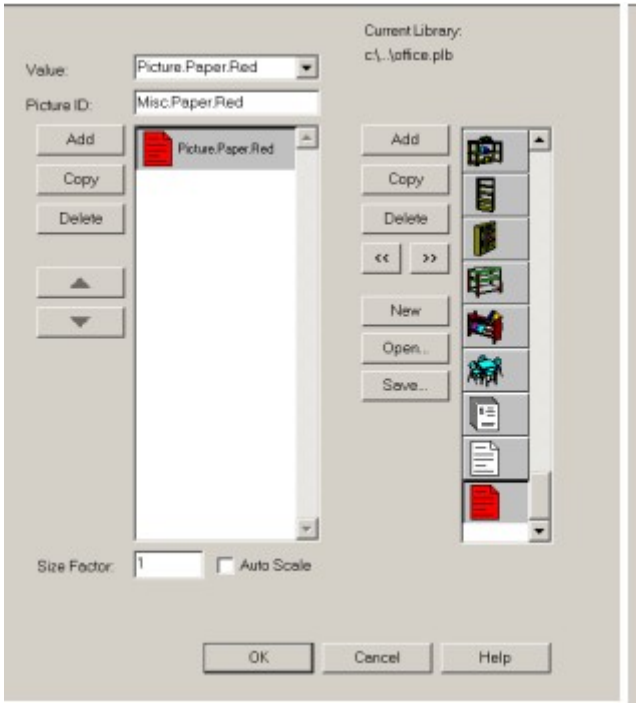


Рисунок 13.21 – Выбор картинки

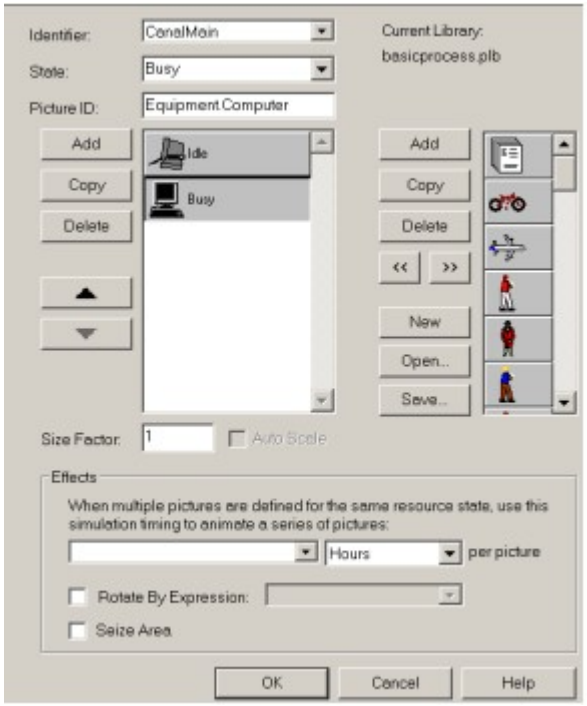


Рисунок 13.22 – Окно настроек картинок

Для вставки новой картинки из раскрывающегося списка нужно выбрать ее в списке справа, затем нажать кнопку < <

Картинка появится в центральной области окна (рисунок 13.21), она должна быть в активном (выделенном состоянии), в противном случае ее нужно выделить. Pictore ID уже будет у выделенной картинки, а вот Value нужно придумать и ввести, следуя правилам. В предложенном варианте это имя – Picture.Paper.Red. Далее следует нажать Ok. Вы вернетесь в область диаграммы. Там следует выбрать место (произвольно), где в области диаграммы разместится ваша новая картинка (курсор имеет вид +), и вставить ее, нажав правую кнопку мыши. Можно также изменить ее размер.

На рисунке 13.20 вставлена картинка (красный лист) под блоком CreateMessage. Аналогично вставлена картинка (болты) над блоком Create Trouble. Но это еще не все. Для присвоения каждому типу Entity новой картинки следует зайти в окно невизуального модуля свойств Entity на панели Basic Process (рисунок 13.20).

В свойстве Initial Picture выбрать из раскрывающегося списка имя, которое вы ранее дали картинке. Если имени не будет в списке, его нужно ввести вручную, удалив имя по умолчанию.

При нажатии на кнопку  появляется окно настройки картинок для состояний ресурсов, представленное на рисунке 13.22. В центральной части могут отображаться различные картинки, оставшиеся от предыдущей работы, их можно просто удалить Delete. Каждому ресурсу можно присвоить различные картинки для состояний Idle (свободен) и

Busy (занят). В этом случае следует добавить эти картинки с помощью кнопки << в центральное поле (рисунок 13.22).

При выделенной картинке в центральном поле заполнить значения параметров Identifier, State, Pictory Id из раскрывающихся списков или ввести их вручную. Это нужно при необходимости сделать для каждого ресурса. При этих действиях картинки автоматически появятся в области диаграммы, их можно перемещать по полю, куда желательно. На рисунке 13.22 состояния ресурса CanalMain отображаются в виде различных расположений монитора. Заметим, что двойным щелчком по картинке на области диаграммы (рабочем поле) вызывается именно те окна (рисунки 13.21, 13.22), которые настроены на эти картинки. При нажатии кнопок и будут отображаться эти же окна, но в ненастроенном виде.

Пример 4. Запуск модели на выполнение.

Параметры модели в целом задаются в пункте главного меню Run-Setup (рисунок 13.23), вкладка Replication Parameters. Здесь, как минимум, нужно задать параметры Replication Length и Base Time Unit в соответствии с условием задачи.

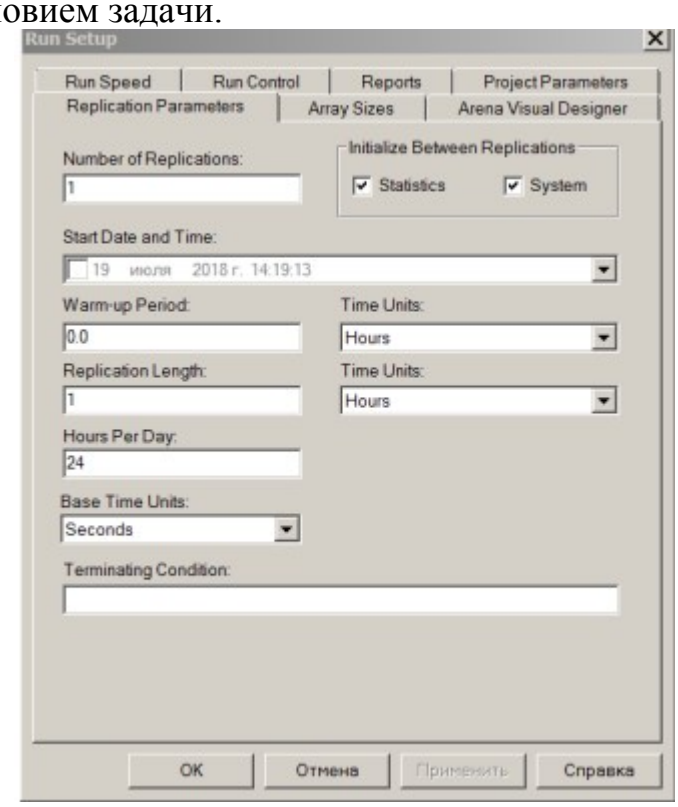


Рисунок 13.23 – Окно Run- Setup для установки параметров модели

Запускается модель с помощью специальной панели инструментов, изображенной на рисунке 13.24.

Рисунок 13.24 – Панели инструментов для запуска модели

Эта панель позволяет запускать, приостанавливать модель, прокручивать ее без анимации, регулировать скорость проигрывания, выполнять проигрывание по шагам.

В результате прогона созданной модели получились результаты, представленные в рабочей области диаграммы (рисунок 13.25): количество сообщений, переданных по основному каналу, равно 359, по резервному – 37, потерянных сообщений нет, количество сбоев равно 16.

Изменив исходные данные, можно снова запустить модель на выполнение и посмотреть, как это отразится на результатах.

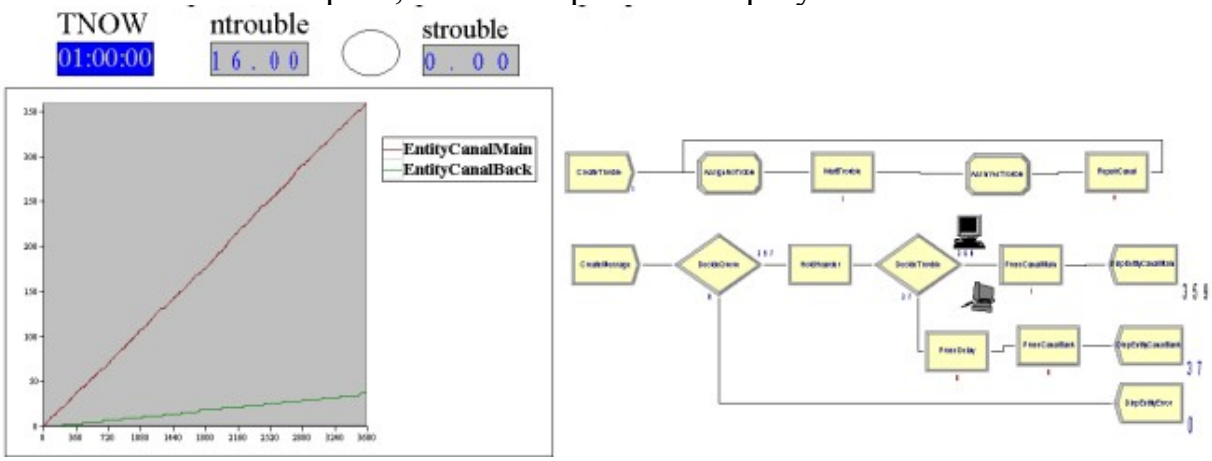
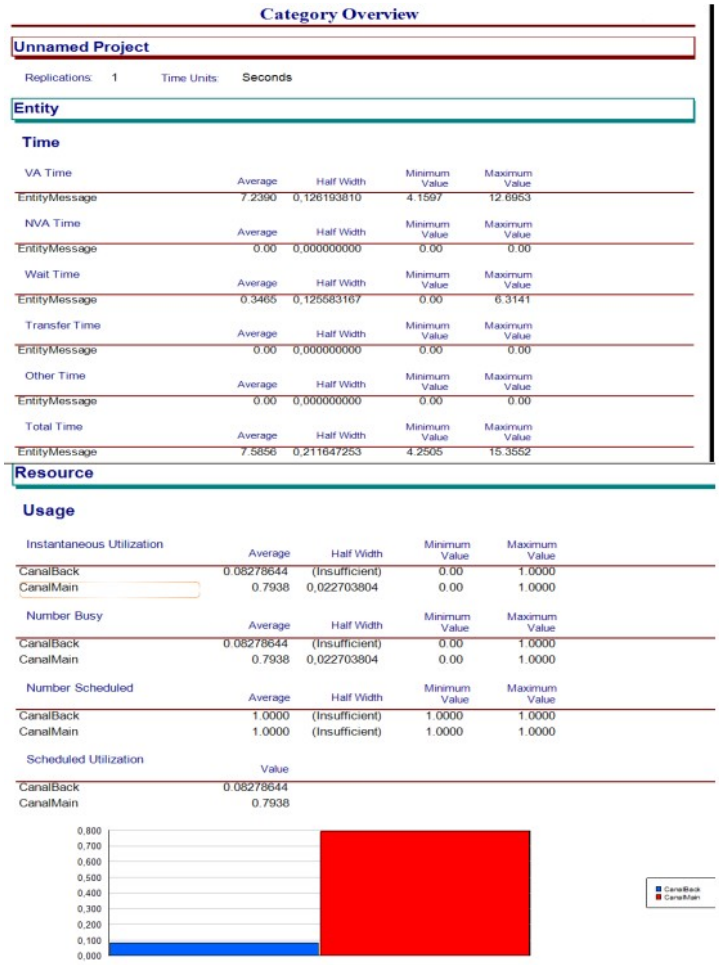


Рисунок 13.25 – Схема модели и результаты моделирования

Пример 5. Отчеты

На панели Reports окна проектов представлены различные виды отчетов, которые автоматически формируются по завершении прогона модели.



Category Overview

Unnamed Project

Replications: 1 Time Units: Seconds

Entity

Time

VA Time	Average	Half Width	Minimum Value	Maximum Value
EntityMessage	7.2390	0,126193810	4.1597	12.6953

NVA Time	Average	Half Width	Minimum Value	Maximum Value
EntityMessage	0.00	0,000000000	0.00	0.00

Wait Time	Average	Half Width	Minimum Value	Maximum Value
EntityMessage	0.3465	0,125583167	0.00	6.3141

Transfer Time	Average	Half Width	Minimum Value	Maximum Value
EntityMessage	0.00	0,000000000	0.00	0.00

Other Time	Average	Half Width	Minimum Value	Maximum Value
EntityMessage	0.00	0,000000000	0.00	0.00

Total Time	Average	Half Width	Minimum Value	Maximum Value
EntityMessage	7.5856	0,211647253	4.2505	15.3552

Resource

Usage

Instantaneous Utilization	Average	Half Width	Minimum Value	Maximum Value
CanalBack	0.08278644	(Insufficient)	0.00	1.0000
CanalMain	0.7938	0,022703804	0.00	1.0000

Number Busy	Average	Half Width	Minimum Value	Maximum Value
CanalBack	0.08278644	(Insufficient)	0.00	1.0000
CanalMain	0.7938	0,022703804	0.00	1.0000

Number Scheduled	Average	Half Width	Minimum Value	Maximum Value
CanalBack	1.0000	(Insufficient)	1.0000	1.0000
CanalMain	1.0000	(Insufficient)	1.0000	1.0000

Scheduled Utilization	Value
CanalBack	0.08278644
CanalMain	0.7938

Рисунок 13.26 – Окно отчета

Содержания различных отчетов во многом пересекаются, поэтому можно ограничиться обзорным отчетом Category Overview, который обычно состоит из нескольких страниц. Фрагменты этого отчета представлены на рисунке 13.26. Здесь приводятся значения (средние, минимальные, максимальные) системных переменных, каковыми являются время пребывания Entity в системе, время ожидания в очереди, коэффициенты использования ресурсов, количество сущностей, участвующих в различных процессах и др.

Аппаратура и материалы. Для выполнения лабораторной работы необходимо использовать следующее: аппаратное обеспечение: персональный компьютер и программное обеспечение: операционную систему Windows 10 и выше; Microsoft Office13 и выше, среда имитационного моделирования ARENA.

Указания по технике безопасности. Студенты должны следовать общепринятой технике безопасности для пользователей персональных компьютеров. Не следует самостоятельно производить ремонт технических средств, установку и удаление программного обеспечения. В случае обнаружения неисправностей необходимо сообщить об этом администратору компьютерного класса (обслуживающему персоналу лаборатории).

Методика и порядок выполнения работы

Выполните предложенные задания, предварительно рассмотрев приведенные в примеры.

Задание 13.1. Построить модель «Магистраль передачи данных».

Задание 13.2. Настроить параметры модели.

Задание 13.3. Настроить анимацию модели.

Задание 13.4. Провести вычислительный эксперимент с моделью.

Задание 13.5. Вывести отчеты результатов моделирования и выполнить анализ, полученных результатов.

Содержание отчета и его форма

Подготовьте отчет, в котором приведите технологию выполнения заданий.

Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) формулировку задания и технологию его выполнения;
4) ответы на контрольные вопросы;
5) приложение – файлы выполненных заданий.

Вопросы для защиты работы

1. Для чего используется среда имитационного моделирования ARENA?
2. Как осуществляется моделирование в ARENA?
3. На моделирование каких систем ориентирована среда ARENA?
4. Какое основное преимущество среды ARENA по сравнению с аналогичными системами имитационного моделирования?
5. С помощью среды ARENA можно моделировать систем массового обслуживания?

ЛАБОРАТОРНАЯ РАБОТА 14
Моделирование информационных процессов в среде AnyLogic

Цель и содержание: изучить основные приемы работы со средой имитационного моделирования AnyLogic и приобрести навыки построения моделей в этой среде.

Организационная форма занятий: решение проблемных задач, разбор конкретных ситуаций

Вопросы для обсуждения на лабораторном занятии: построение моделей в среде имитационного моделирования AnyLogic.

Формируемые компетенции: ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2

Теоретическое обоснование

Среда имитационного моделирования AnyLogic основана на Java и базируется на платформе Eclipse – современном стандарте для бизнес-приложений. Благодаря Eclipse AnyLogic работает на всех распространенных операционных системах (Windows, Mac, Linux и т.д.).

Рассмотрим решение разобранной в лабораторной работе задачи «Магистраль передачи данных» средствами AnyLogic.

Логику решения задачи сохраним, будем использовать те инструменты разработки Anylogic, которые подобны рассмотренным ранее.

При создании более сложных моделей могут возникнуть сложности с описанием логики задачи, поскольку нужно использовать конструкции на языке Java. Тем, кто владеет этим языком, безусловно, будет проще. Однако, предоставляемые справочные материалы позволяют преодолеть эти трудности и тем, кто имеет о Java весьма поверхностное представление.

Для выполнения заданий по моделированию в системе AnyLogic нужно скачать и установить на компьютере свободно распространяемую версию AnyLogic PLE (время ее использования не ограничено) с официального сайта разработчика https://www.anylogic.ru/downloads/ или по ссылке https://www.anylogic.com/files/anylogic-ple-8.2.3.x86_64.exe.

При первоначальном запуске программы Anylogic отображается Начальная страница, на которой пользователю предлагается ознакомиться с многочисленными примерами моделей, разработанных в этой системе. Попасть на эту страницу в ходе дальнейшей работы можно из меню Справка. Нужно сказать, что даже в бесплатной версии AnyLogic разработчики предоставляют огромное количество материалов для обучения.

На рисунке 14.1 показан интерфейс среды имитационного моделирования AnyLogic. Сверху располагаются Меню, панели инструментов, по центру – рабочая область (графический редактор или область структурной диаграммы), справа – окно свойств объектов, слева –

Окно панелей Проекты и Палитра, эти панели переключаются щелчком по заголовку. В нижней части размещаются окно ошибок и др. Отображение тех или иных окон можно настроить с помощью кнопки Вид в главном меню AnyLogic. Окно панели Проекты (слева), навигатор системы, отображает имена проектов, открываемых в последнее время. При ненадобности их можно закрыть и окно станет пустым. Каждый проект организован иерархически в виде дерева: сам проект образует верхний уровень, эксперименты, классы активных объектов и Java классы образуют следующий уровень. Элементы, входящие в состав активных объектов, вложены в соответствующую подветвь дерева класса активного объекта и т. д.

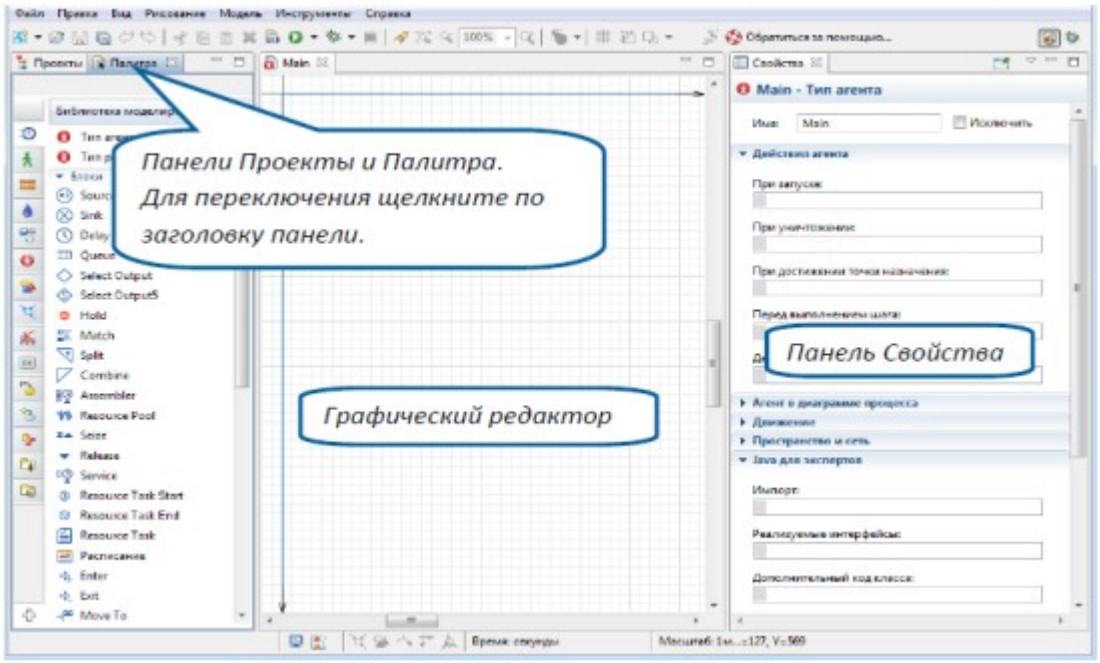


Рисунок 14.1 – Интерфейс среды имитационного моделирования AnyLogic

Окно панели Палитра (слева) включает элементы (графические объекты), которые могут быть добавлены на структурную диаграмму активного объекта. Элементы разбиты по группам, отображаемым на разных вкладках Палитры. Количество отображаемых вкладок можно увеличить (уменьшить), используя «+» в нижнем левом углу окна Палитра. Вкладки переключаются по щелчку. Объекты палитры добавляются на диаграмму модели путем перетаскивания их в окно графического редактора.

Пример 1. Начало работы в среде имитационного моделирования AnyLogic.

Для создания модели выберите в главном меню AnyLogic: Файл > Создать > Модель. Откроется диалоговое окно Новая модель (рисунок 14.1), в котором нужно задать имя модели, в нашем случае это Магистраль, указать путь сохранения модели или принять тот путь,

который создается по умолчанию. В нашем случае путь изменен. Созданный проект не забывайте периодически сохранять по ходу наполнения данными.

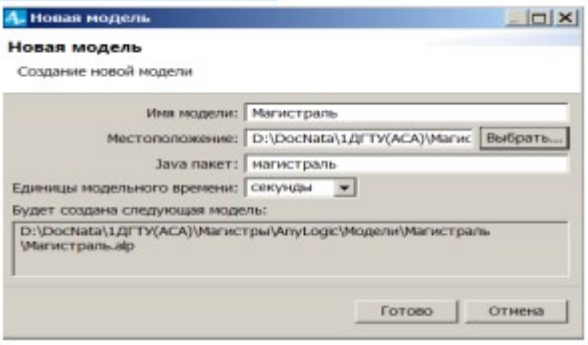


Рисунок 14.2– Окно диалога Новая модель

В результате создания модели по сути первоначально создается структура проекта в виде иерархического дерева, которую можно увидеть, открыв окно панели Проекты.

В окне Проекты может быть несколько открытых моделей, на рисунке 14.3 их две: Магистраль3 и Магистраль. Модель Магистраль только что создана, пока не заполнялась данными, на ней можно проиллюстрировать структуру вновь создаваемого проекта. По умолчанию в каждой модели создается один тип агента – Main и один простой эксперимент Simulation: Main, содержащий настройки запуска этой модели, встроенную Базу данных и Презентацию.

Для создания в модели новых типа агента, эксперимента, базы данных, Java-класса и др. нужно в окне Проекты щелкнуть по имени модели правой кнопкой, далее в меню Создать выбрать из раскрывающегося списка создаваемый класс.

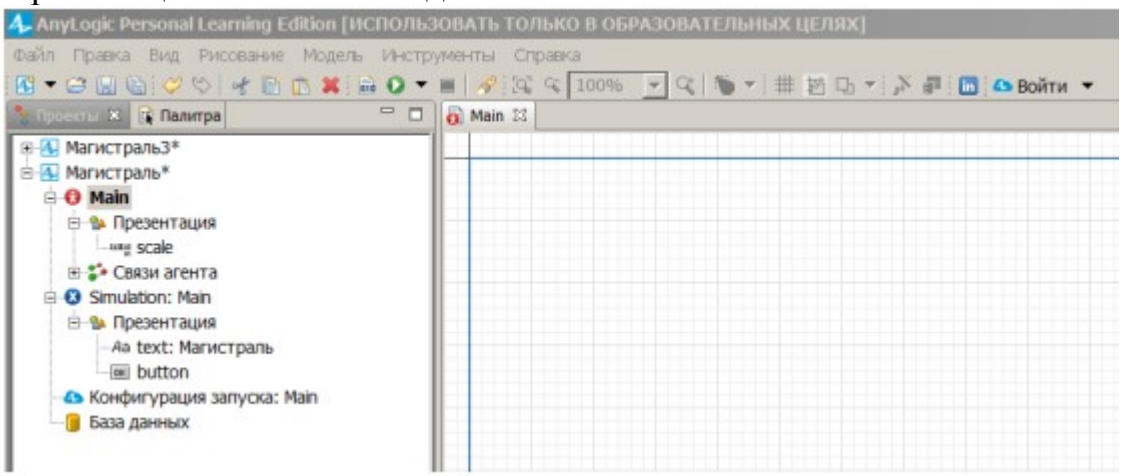


Рисунок 14.3– Окно Проекты с двумя открытыми моделями Магистраль3 и Магистраль

Если дважды щелкнуть по иконке 

в окне Проекты, то по центру открывается графический редактор диаграммы агента верхнего уровня Main (рисунок 14.3). Это главное графическое окно проекта, в нем

создается структурная схема, размещается анимационная картинка, которая отображается при проигрывании модели. В принципе, имя Main можно заменить на другое, но тогда так же нужно изменить имя простого эксперимента.

Следует заметить, что термин «Агент» в AnyLogic используется в широком смысле и применяется к различным по смыслу и действиям объектам.

Здесь в качестве агентов могут быть транзакты и ресурсы, но, кроме этого, узлы иерархического дерева, проекты, идеи и т.д. ***Пример 2.*** Диаграмма процессов модели.

Чтобы упростить себе задачу освоения AnyLogic, не будем менять логику модели, описанной при разработке модели Магистрали в среде Arena. 27 Пусть диаграмма процессов, как и раньше, имеет две ветви, соответствующие сущностям Message и Trouble. Обе эти ветви нужно разместить в окне графического редактора агента Main. Кроме того, в Anylogic сущности Message и Trouble, как и Main, должны являться агентами верхнего уровня.

На рисунке 14.4 показана уже созданная диаграмма агента Main для модели Магистраль4 в окне графического редактора (вызывается окно двойным щелчком по иконке в окне Проекты).

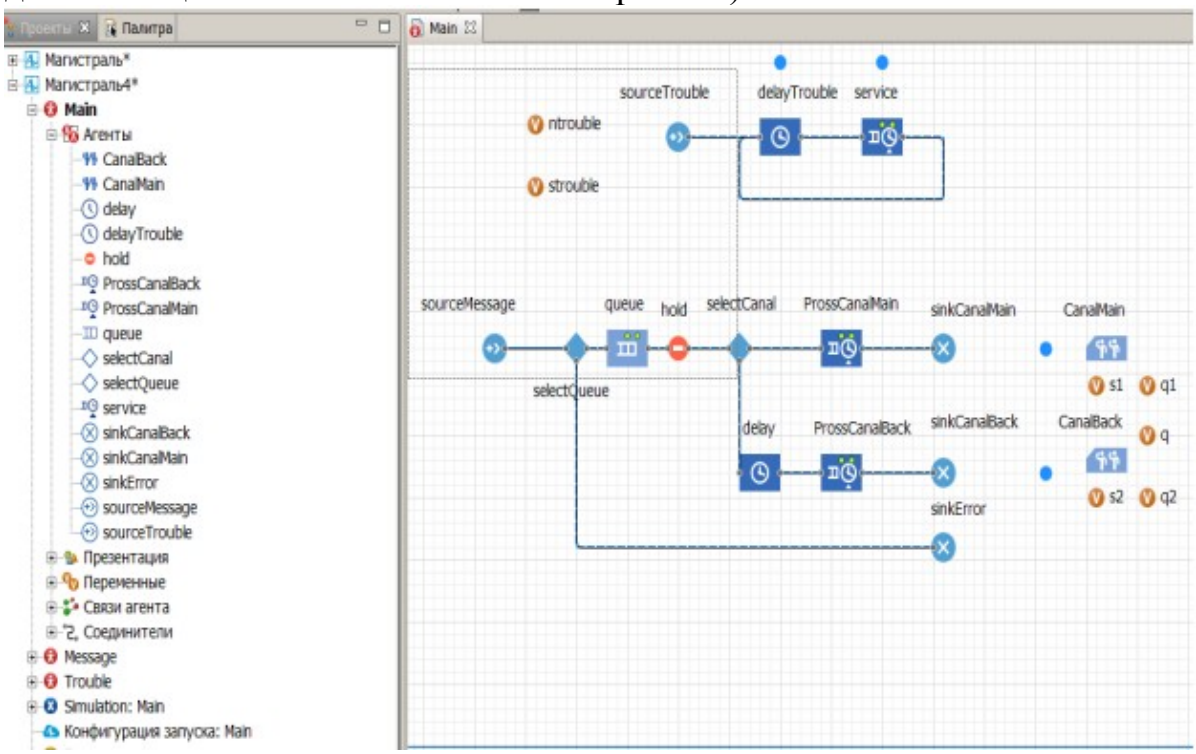


Рисунок 14.4– Графический редактор диаграммы агента верхнего уровня Main, с созданной диаграммой агента Main для модели Магистраль4

Модули, используемые в диаграмме AniLogic, подобны тем, которые были в Arena. Однако, здесь отсутствуют модули типа Assign, которые предназначены для присваивания значений переменным и параметрам. В AniLogic это делается в окне свойств в полях Действия.

Опишем этапы создания диаграммы. Откройте пустое окно графического редактора агента Main в создаваемой модели, дважды щелкнув по иконке Main в окне Проекты. Переключитесь на панель Палитры, откройте в ней Библиотеку моделирования процессов, если она сразу не открылась, щелчком по иконке (рисунок 14.5). Для добавления объекта на диаграмму надо щёлкнуть его мышью и перетащить в графический редактор. При перетаскивании объектов автоматически появляются соединительные линии между объектами. Эти линии должны соединять только порты, находящиеся с правой или левой стороны иконок. Если не получится автоматическое соединение нужных объектов, дважды щёлкните начальный порт. Он станет зелёным. Протащите курсор к конечному порту. Он также станет зелёным. Для завершения процесса соединения щёлкните конечный порт.

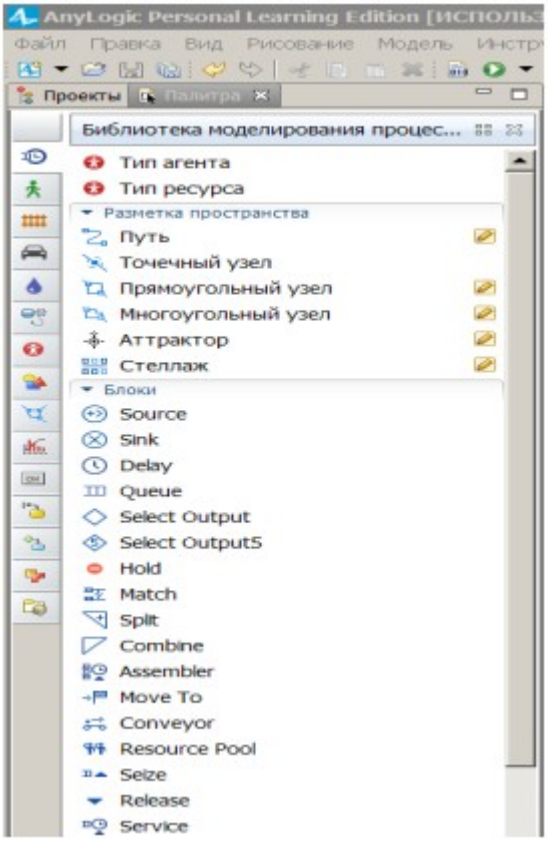


Рисунок 14.5– Библиотека моделирования процессов

Пример 3. Описание моделирования сбоя в канале связи.

На рисунке 14.6 показана ветвь диаграммы процессов, соответствующая агенту Trouble, создающему сбой в системе, которую нужно создать. Здесь использованы объекты Source (вход), Delay (задержка), Service (захват-задержка-освобождение ресурса). Свойства объекта (как и любого другого элемента AnyLogic) можно изменить в панели Свойства, расположенной справа. Для изменения свойств объекта

нужно будет предварительно щелчком мыши выделить его в графическом редакторе или в панели Проекты.

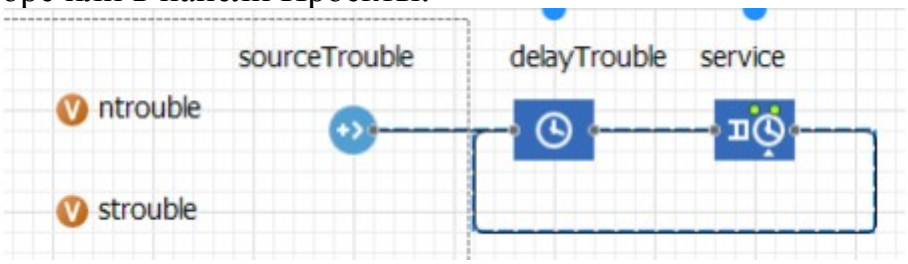


Рисунок 14.6 – Ветвь диаграммы процессов, соответствующая агенту Trouble, создающему сбой в системе

Создайте диаграмму процессов (синие блоки) без настройки свойств, как показано на рисунке 14.6.

На рисунке 14.7 показано уже настроенное окно свойств объекта sourceTrouble в соответствии с принятой логикой задачи.

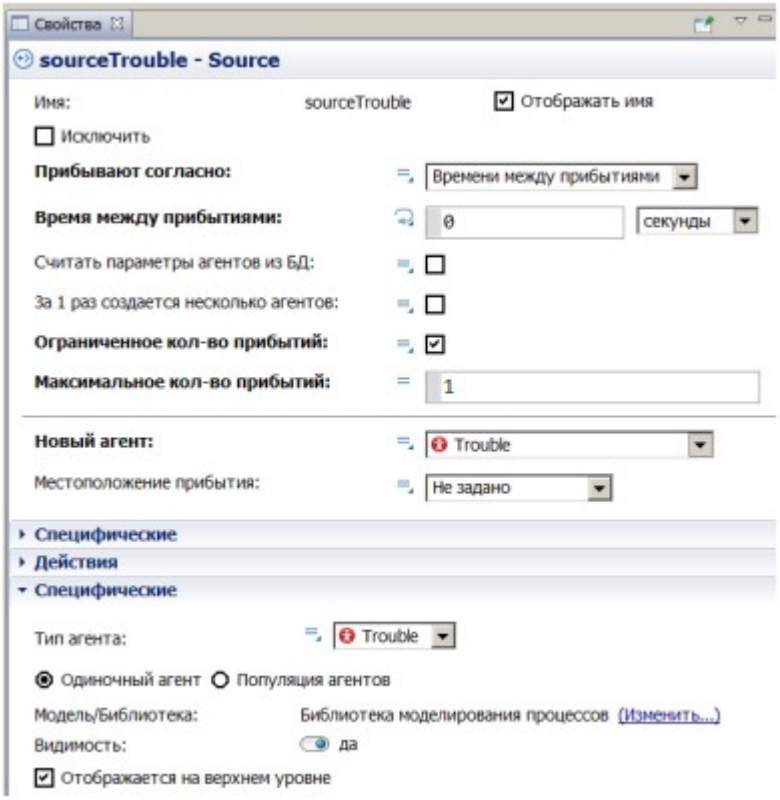


Рисунок 14.7 – Окно свойств объекта sourceTrouble

Чтобы вызвать это окно, щелкните по объекту в графическом редакторе. Заполните окно по образцу. Первоначально в поле Новый Агент (рисунок 14.8) отображается стандартное имя Агент. Здесь нужно выбрать создать другой тип .

Рисунок 14.8 – Поле Новый Агент со стандартным именем Агент

Появится окно мастера (рисунок 35) создания агентов, в котором следует задать имя нового агента Trouble и далее следовать указаниям.

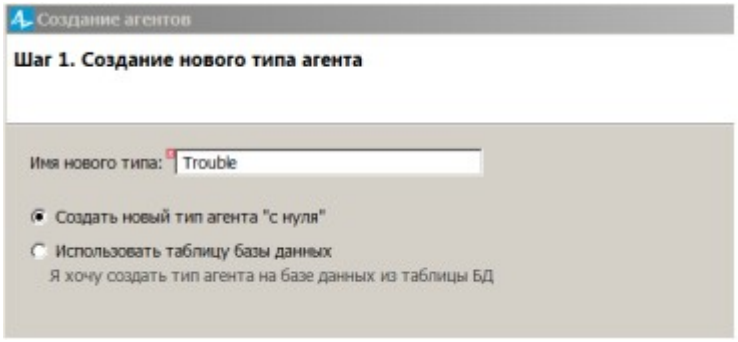


Рисунок 14.9 – Окно мастера создания агентов

Для вновь созданного агента Trouble создается новое окно графического редактора. Если в ходе создания агента задавалась его анимационная картинка, то она будет отображаться в этом окне. При создании более сложных моделей в этом окне располагаются переменные и параметры агента и т.д.

Окно мастера создания агентов можно вызвать из окна Проекты, правой кнопкой мыши щелкнув по имени проекта, в котором создается новый агент, и выбрать Создать Тип агента. Для продолжения работы нужно снова переключиться на окно Main в Графическом редакторе. Не забываем периодически сохранять модель.

На рисунке 14.10 показано уже настроенное окно свойств объекта delayTrouble в соответствии с условием задачи. Заполните это окно по образцу.

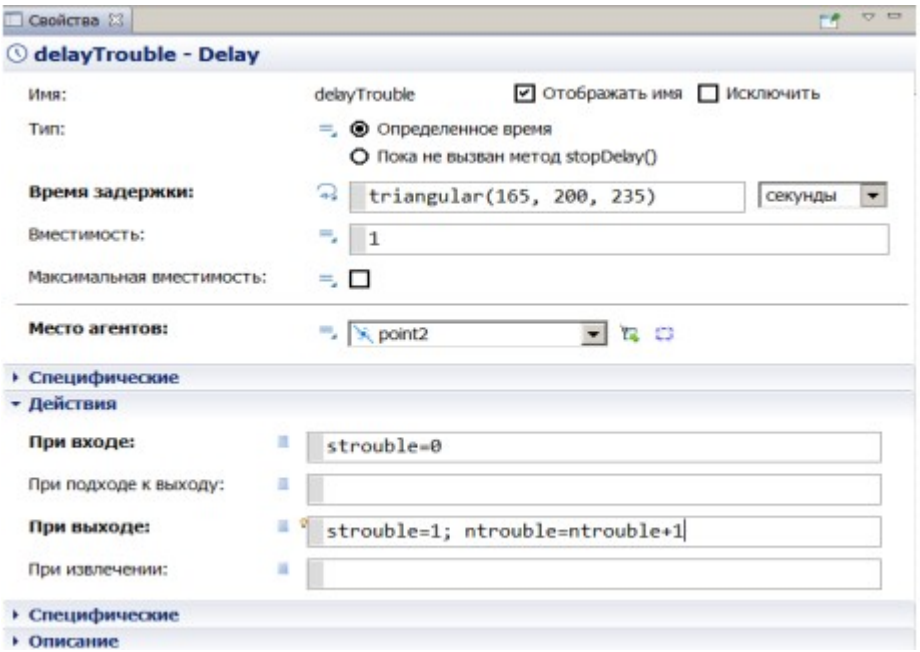


Рисунок 14.10 – Окно мастера создания агентов

Для времени задержки выбран треугольный закон. В AnyLogic возможно задать и другие стандартные законы распределения. Рассмотрим как заполнить поле Время задержки. Чтобы не набирать вручную функцию triangular(165, 200, 235), а выбрать ее из раскрывающегося списка, подведите курсор в начало поля ввода функции и нажмите комбинацию Ctrl+пробел (выпадающий список может появиться спустя некоторое время). Выберите из списка нужную функцию и измените ее аргументы в соответствии с условием. Обратите внимание, что в AnyLogic аргумент λ показательного распределения exponential (λ) – это интенсивность потока транзактов, т.е. величина, обратная среднему времени между прибытиями транзактов. Так, если транзакты прибывают в среднем раз в 10 минут, то надо указывать распределение exponential (0.1) . Заметим, что в Арене надо было писать exponential (10).

Место агентов point2 оставьте незаполненным. Это поле может быть заполнено только после добавления элементов разметки пространства на диаграмму для настройки анимации. В данной модели анимацию пока не будем создавать.

В свойстве Действия (здесь записываются операторы на Java) задайте значения переменным сбоя в канале CanalMain и числа сбоев (strouble, ntrouble), как показано на рисунке 14.10. Но этого для дальнейшей работы недостаточно. Переменные должны быть предварительно создана. Это делается следующим образом: на панели Палитра выбрать вкладку Агент, щелкнув по иконке  . Среди компонент агента (рисунок 14.11) следует выбрать компоненту  Переменная, перетащить ее в область диаграммы агента Main и расположить в каком-либо месте (рисунок 14.4).

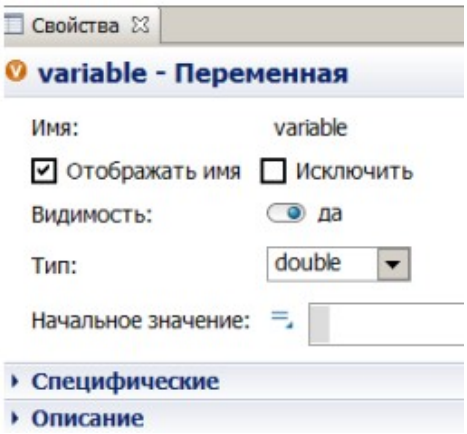
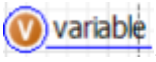


Рисунок 14.12– Окно свойств переменной

Далее нужно щелкнуть по объекту в области диаграммы агента Main 

.

В появившемся справа окне свойств

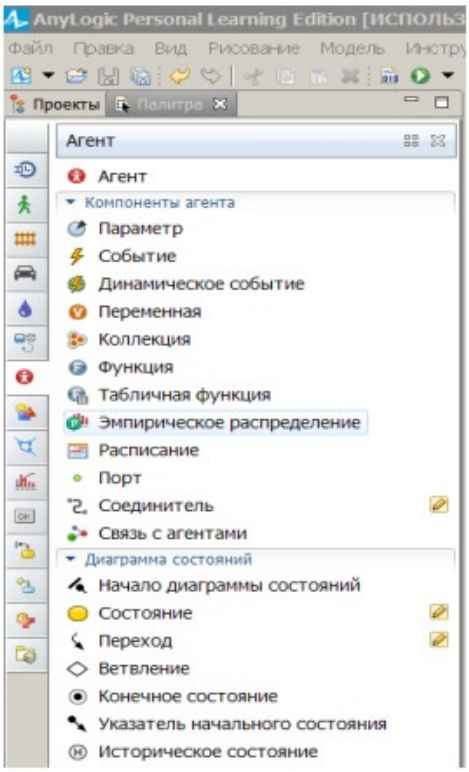


Рисунок 14.11– Выбор среди компонент агента необходимую компоненту

(рисунок 14.12) присвоить переменной имя strouble, тип и начальное значение 0.

Аналогично создайте переменную ntrouble. Не забывайте периодически сохранять модель

На рисунке 39 показано уже настроенное окно свойств объекта service в соответствии с условием задачи. Чтобы вызвать это окно, щелкните по объекту. Чтобы выбрать Тип ресурсов (из раскрывающегося списка), нужно предварительно эти ресурсы создать. В строке Количество ресурсов нужно указать, сколько ресурсов задействованы в процессе service для обслуживания транзакта.

Опишем, как создаются ресурсы. В нашем случае это ресурсы CanalMain и CanalBack. В библиотеке моделирования процессов выбираем объект Resource Pool, перетаскиваем его в графическую область диаграммы агента Main. В графической области появится объект  , размещаем егов произвольном месте.

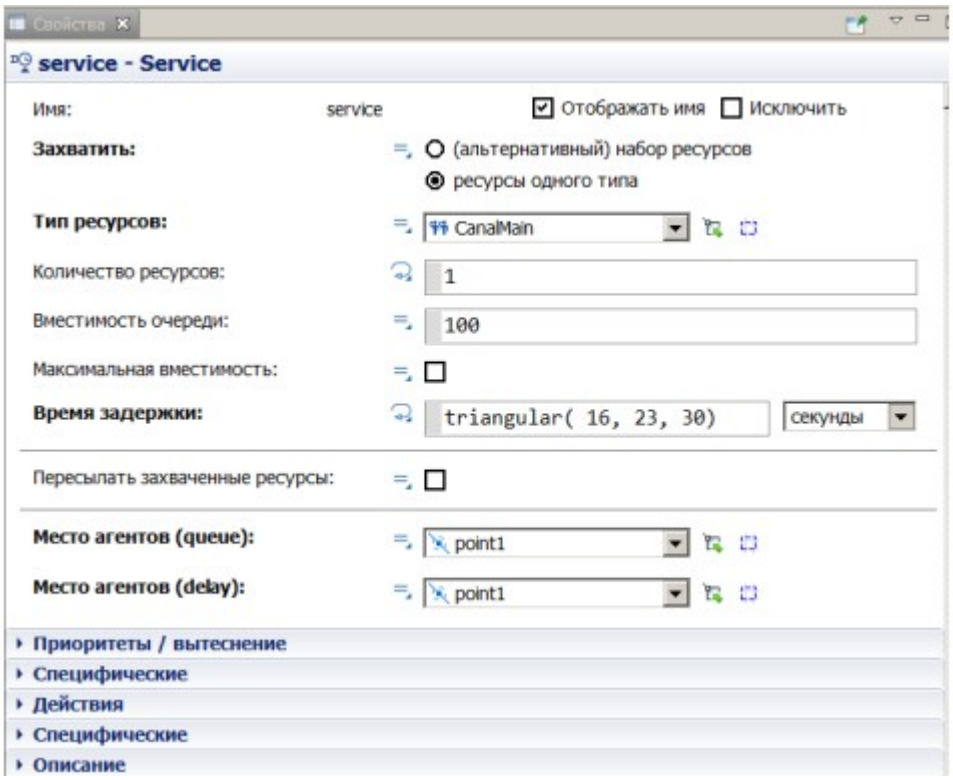


Рисунок 14.13 – Настроенное окно свойств объекта service

Щелчком по этому объекту вызываем окно свойств ресурса (рисунок 14.14), в котором вводим имя CanalMain и количество ресурсов – 1 (в этом случае это его мощность). Если имеется несколько ресурсов данного типа, например, несколько компьютеров или несколько рабочих для параллельного обслуживания транзакта, то нужно указывать это число. В окне ресурса можно добавить новый ресурс-агент, но мы этого не будем делать, в поле Новый ресурс оставляем стандартное имя Агент.

Аналогично создаем второй ресурс CanalBack, он понадобится во второй диаграмме.

Поля Место агентов, которые нужны для анимации, пока не заполняем.

Поскольку рассматриваемая ветвь диаграммы процессов циклическая, нужно соединить выходной порт объекта service с входным портом объекта delayTrouble. Это можно сделать с помощью создания дополнительных узлов (по щелчку мыши) на коннекторе.

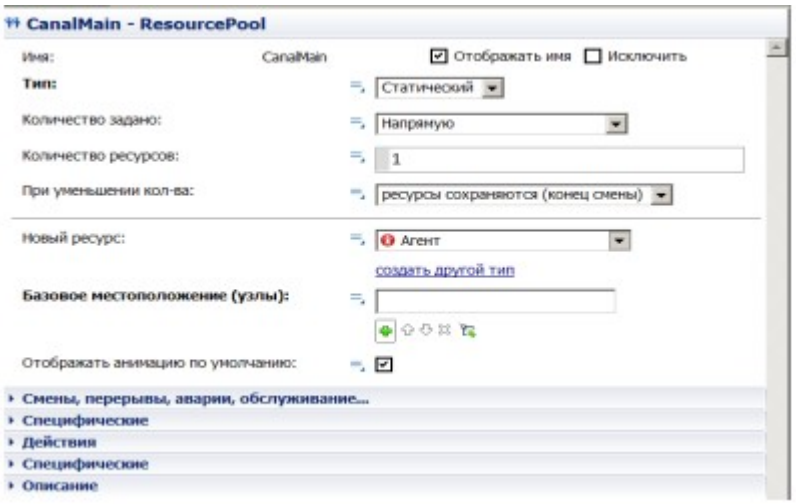


Рисунок 14.14 –

Пример 4. Описание моделирования передачи сообщений.

На рисунке 14.15 показана ветвь диаграммы процессов, соответствующая агенту Message, которую нужно создать. Создайте ее, как показано на рисунке, но пока без настройки свойств.

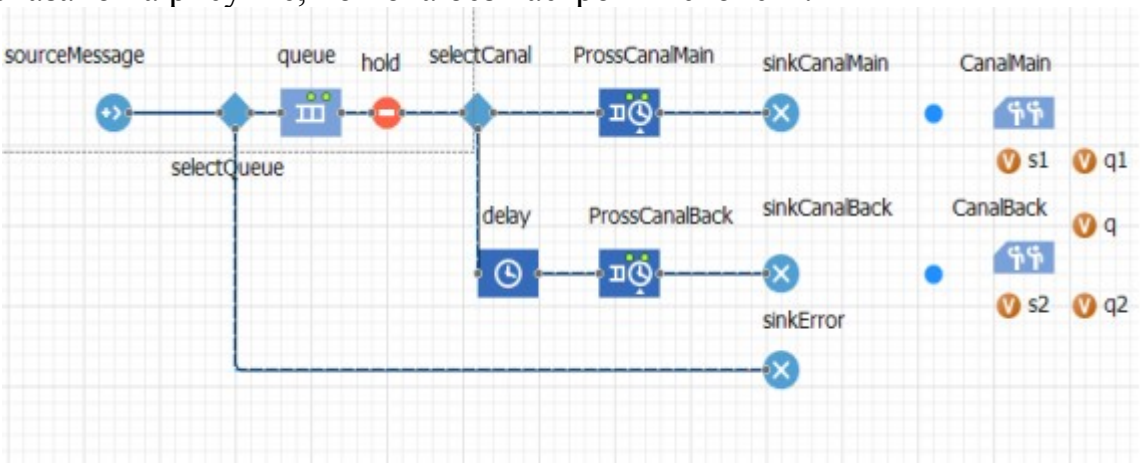


Рисунок 14.15 – Ветвь диаграммы процессов, соответствующая агенту Message

На рисунке 14.16 представлено уже заполненное окно свойств объекта sourceMessage.

В AnyLogic, в отличие от Arena, блок Hold не имеет настроек очереди, хотя он способен ее создавать. Для определения длины очереди нужно перед блоком Hold разместить объект Queue.

На рисунке 14.17 представлено окно свойств объекта queue. Оставьте допустимую вместимость очереди равной 100.

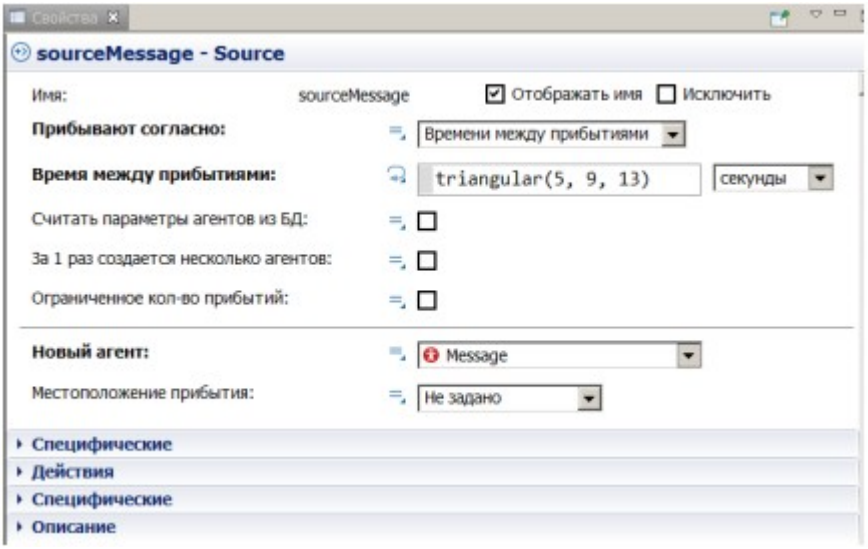


Рисунок 14.16 – Окно свойств ресурса, заполненное для объекта sourceMessage

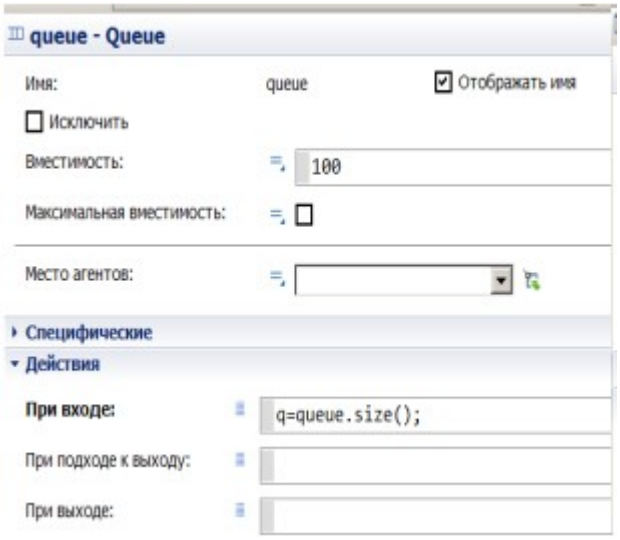


Рисунок 14.17 – Окно свойств объекта queue

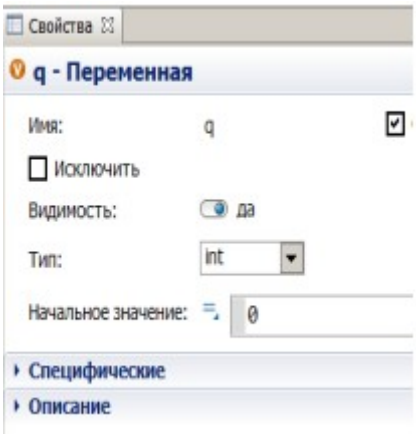
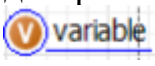


Рисунок 14.18 – Окно свойств переменной *q*

В секции Действия следует переменной *q* – длина очереди, присвоить значение текущей длины очереди с помощью функции size(). Используйте Ctrl+пробел для вызова подсказки или наберите в поле имя функции. Сведения о функциях AnyLogic можно найти в справочных материалах.

Переменная *q* должна быть предварительно создана: на область диаграммы агента Main из вкладки Агент Палитры перетащите иконку 

. В окне свойств поменяйте имя, тип, начальное значение переменной (рисунок 14.18).

В окне свойств объекта Hold (рисунок 14.19) в секции Действия следует задать оператор hold.setBlocked(true) блокировки шлагбаума Hold при занятости ресурсов и режим пропуска сообщений по одному. В условиях используются переменные s1,s2 (создайте эти переменные с

помощью variable), которые отвечают за состояния ресурсов. Значение s1=1 означает, что ресурс CanalMain занят, s1=0 означает что ресурс CanalMain свободен и т.д. Эти переменные будут определены далее в блоках ProssCanalMain, ProssCanalBack.

Переменная stroule отвечает за наличие-отсутствие сбоя в основном канале.

Рисунок 14.19 – Окно свойств объекта Hold

На рисунке 14.18 показано окно свойств объекта selectQueue, который обеспечивает разветвление: сообщения теряются, если длина очереди больше 10.

На рисунке 14.19 показано окно свойств объекта selectCanal, который обеспечивает разветвление: сообщения обрабатываются основным каналом, если сбоя нет.

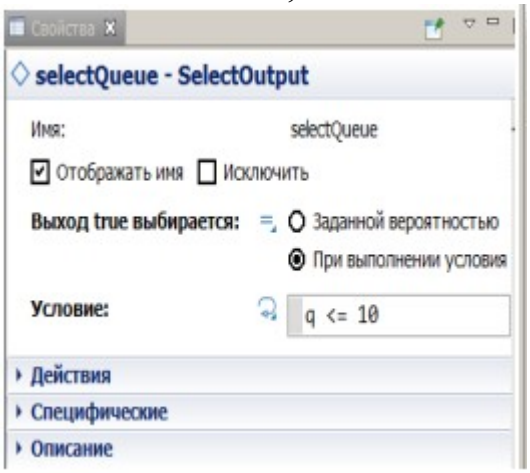


Рисунок 14.20 – Окно свойств объекта selectQueue

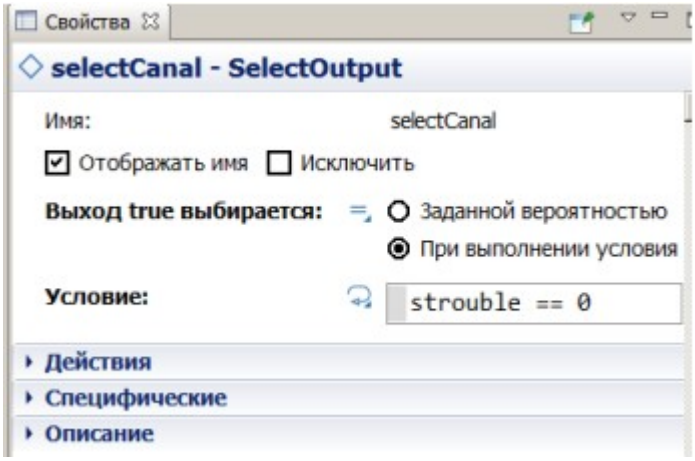


Рисунок 14.21– Окно свойств объекта selectCanal

На рисунке 14.22 показано окно свойств объекта ProssCanalMain. При захвате ресурса присваивается s1=1, при освобождении – s1=0 и записывается оператор разблокировки Hold: hold.setBlocked(false). Здесь также определяется переменная q1 – длины очереди для процесса ProssCanalMain.

Окно свойств объекта ProssCanalBack заполняется аналогично. Окно свойств объекта delay предсталено на рисунке 14.23. В окнах

свойств модулей завершения процессов изменено стандартное имя на имена sinkCanalMain, sinkCanalBack, sinkError (рисунок 14.24).

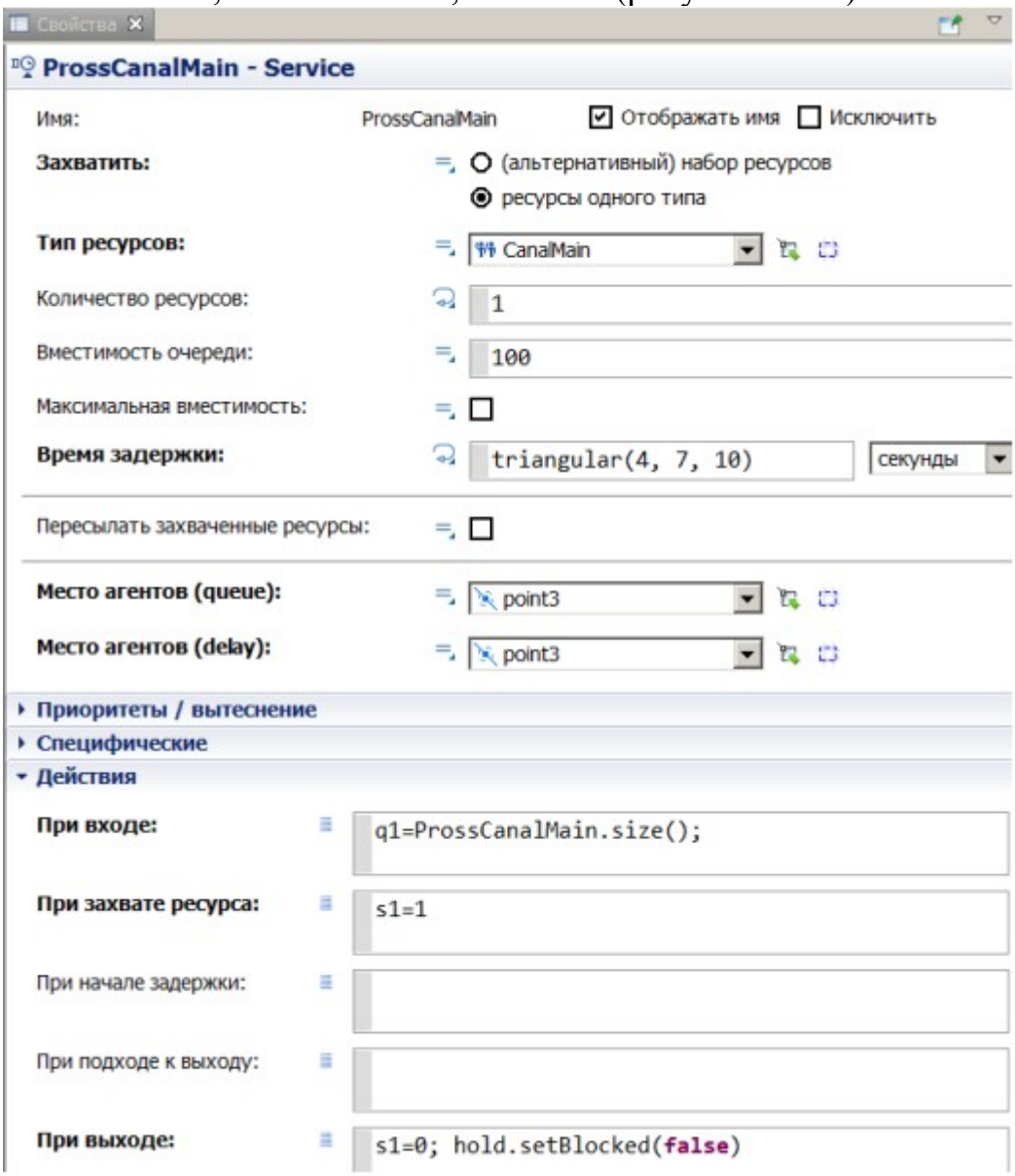


Рисунок 14.22– Окно свойств объекта ProssCanalMain

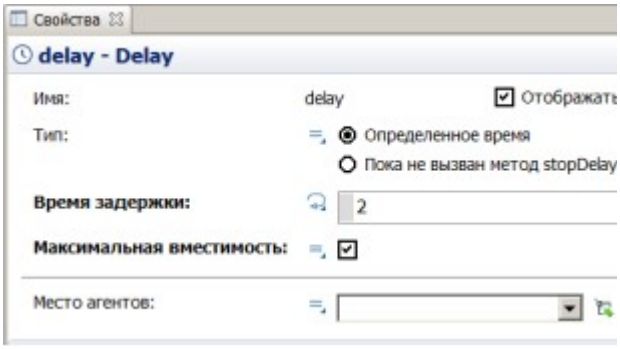


Рисунок 14.23 – Окно свойств объекта Delay

Рисунок 14.24 – Окно свойств объекта sinkCanalMain

Пример 5. Анимация в модели.

В AniLogic предоставляются большие возможности для создания 2D, 3D анимационных моделей. При создании анимации в окне Графического редактора агента Main добавляются компоненты разметки пространства (рисунок 14.25). Элементы разметки в проекте отображаются в окне Проекты в классе – презентации агента. Для материальных агентов можно задать анимационные картинки, которые размещаются на диаграммах этих агентов. Встроенная библиотека картинок доступна в Панели на вкладке 3D объекты. Для создания анимации используются также объекты вкладки Презентация Палитры (рисунок 14.26). Здесь же можно настроить собственные картинки.

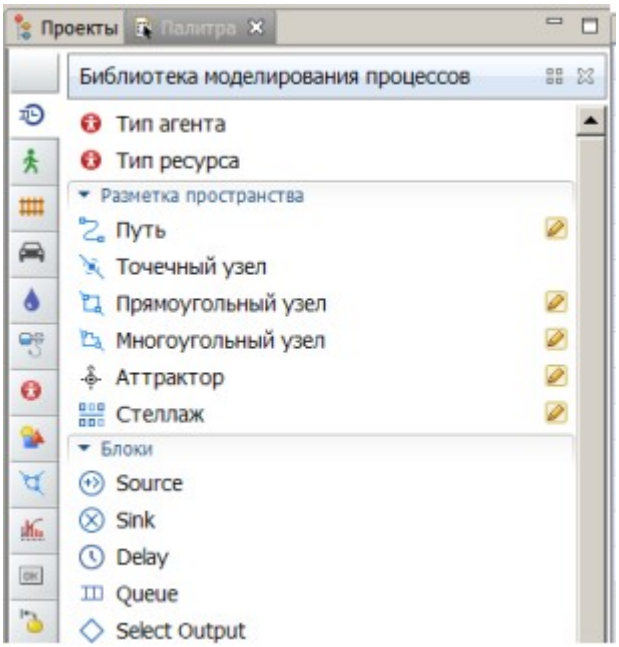


Рисунок 14.25 – Окно Графического редактора агента Main с компонентами разметки пространства

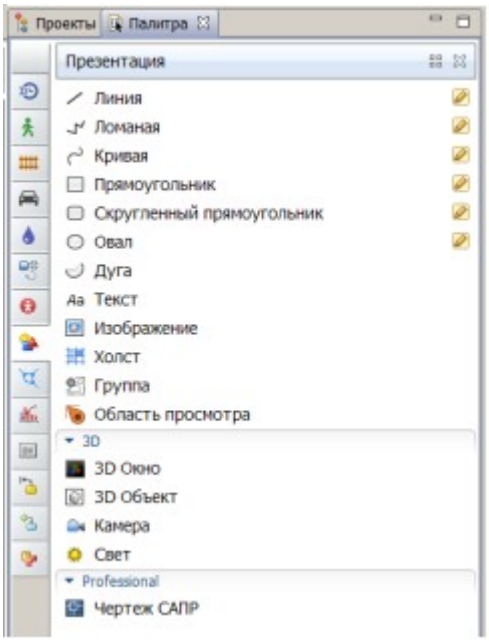


Рисунок 14.26 – объекты презентации вкладки Палитры

В рассматриваемой модели передачи сообщений нет особого смысла создавать дополнительную анимацию. Уже по созданной диаграмме и переменным, размещенным в области диаграммы, при прогоне модели динамика процесса ясна. Можно самостоятельно ознакомиться с соответствующими инструментами и примерам, предоставляемым разработчиками AnyLogic вместе с программным продуктом.

Пример 6. Запуск модели на выполнение.

Прежде чем запускать модель, нужно настроить окно свойств Простого Эксперимента, которое появляется по щелчку по иконке Simulation: Main в окне Проекты (рисунок 14.27). Обязательно нужно выбрать Агент верхнего уровня Main, так как диаграмма процесса создавалась в окне графического редактора именно этого агента. В

настройке, показанной на рисунке 14.27, предполагается, что эксперимент длится в течение 3600 сек. модельного времени. Если не указать этого, то по умолчанию время прогона будет не ограничено.

Если единицы модельного времени должны быть другими: минуты, часы, то нужно в окне свойств Simulation: Main переключиться на вкладку установки модельного времени (внизу окна свойств) и установить нужные единицы модельного времени (рисунок 14.28).

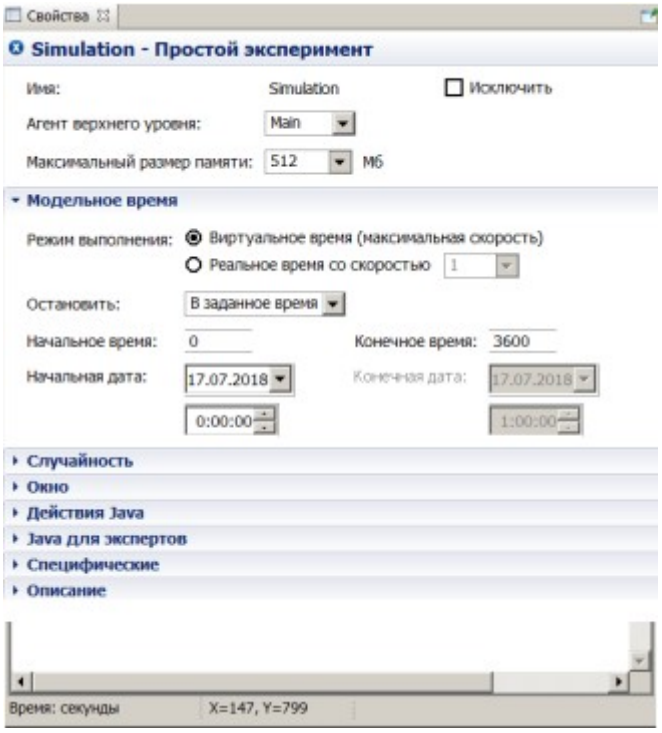


Рисунок 14.27 – Окно свойств Простого Эксперимента

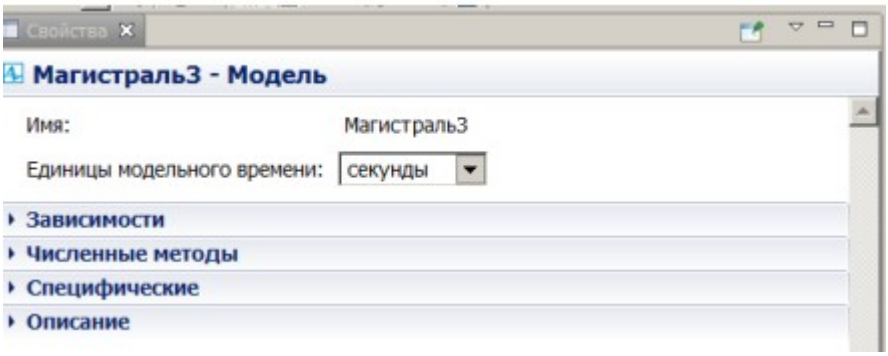
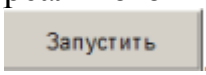


Рисунок 14.28 – Окно с полем выбора единицы модельного времени

Для запуска модели нужно выбрать имя эксперимента в списке, отображаемом при щелчке по черной стрелке на панели инструментов. По умолчанию по зеленой стрелке запускается эксперимент, который запускался предыдущим. После щелчка по стрелке появляется окно презентации эксперимента (рисунок 14.29).

Чтобы регулировать скорость эксперимента, нужно нажать кнопку переключения  с реального на виртуальное время и запустить модель, нажав кнопку 

. Эксперимент можно выполнять в разных режимах, регулируемых с помощью стрелок на панели.

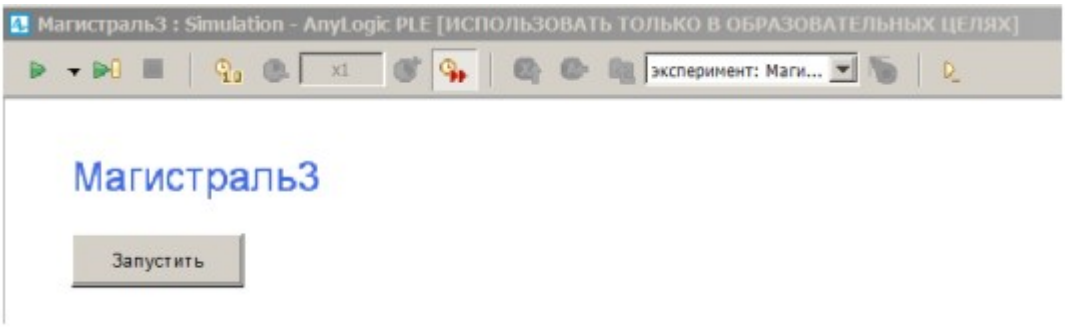


Рисунок 14.29 – Окно презентации эксперимента

На рисунке 14.30 представлены результаты одного прогона с параметрами, определенными в постановке задачи. Здесь использовались анимационные картинки, создание которых подробно не описывалось.

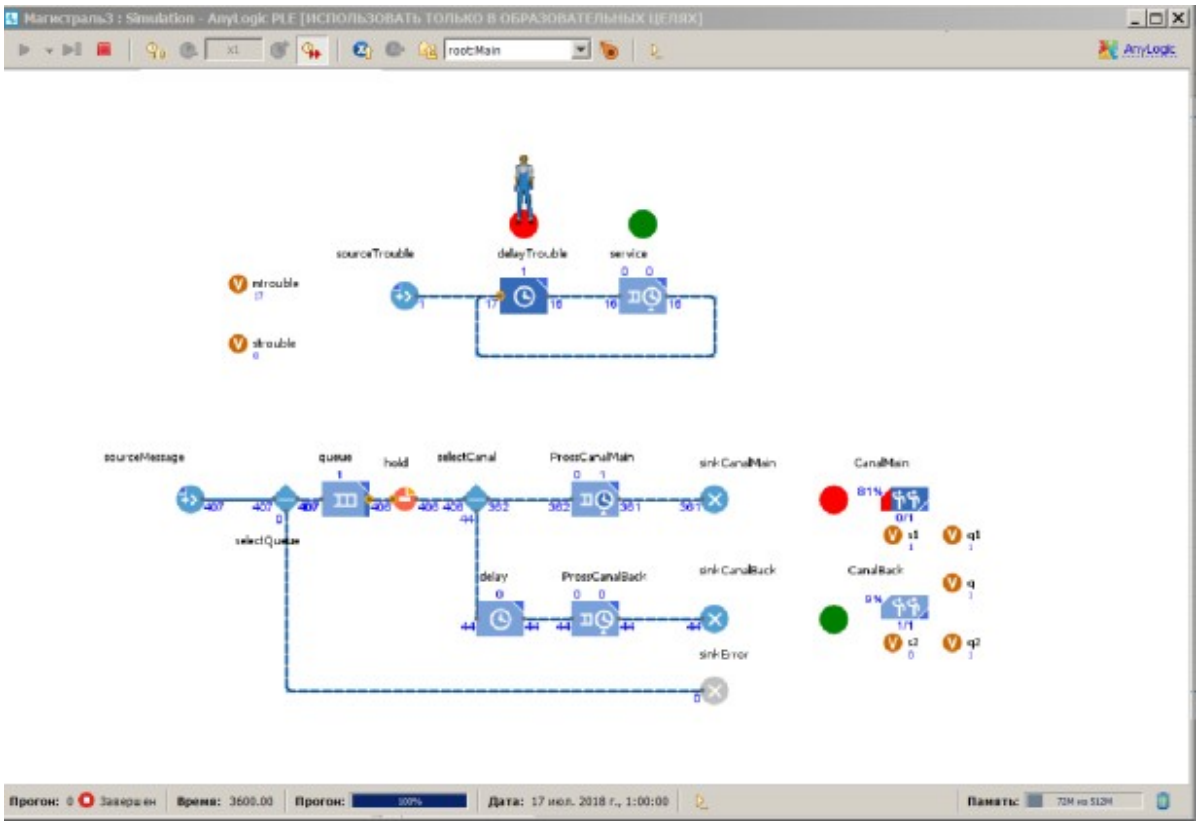


Рисунок 14.30 – Результаты одного прогона с параметрами, определенными в постановке задачи

Сравнение результатов прогонов в Arena и AnyLogic показало, что при заданных в условии параметрах модели результаты почти совпадают.

Однако, изменение входных параметров таким образом, чтобы часть сообщений оказывались потерянными, привело к тому, что результаты стали существенно разниться. Это может объясняться разной настройкой модуля Hold в этих системах, потому что логические условия выбирались одинаково. В AnyLogic транзакты выпускаются из блока Hold по одному при освобождении ресурсов, это прописывается в свойствах модуля, в Arena не ясно как.

В Arena предусмотрены встроенные переменные состояний ресурсов, а в AnyLogic переменные состояний ресурсов определяются пользователем. Все эти факторы, возможно, также влияют на различие результатов.

Аппаратура и материалы. Для выполнения лабораторной работы необходимо использовать следующее: аппаратное обеспечение: персональный компьютер и программное обеспечение: операционную систему Windows 10 и выше; Microsoft Office13 и выше, среда имитационного моделирования AnyLogic.

Указания по технике безопасности. Студенты должны следовать общепринятой технике безопасности для пользователей персональных компьютеров. Не следует самостоятельно производить ремонт технических средств, установку и удаление программного обеспечения. В случае обнаружения неисправностей необходимо сообщить об этом администратору компьютерного класса (обслуживающему персоналу лаборатории).

Методика и порядок выполнения работы

Выполните предложенные задания, предварительно рассмотрев приведенные в теоретическом обосновании примеры.

Задание 14.1. Построить модель «Магистраль передачи данных» в среде имитационного моделирования AnyLogic.

Задание 14.2. Настроить параметры модели.

Задание 14.3. Настроить анимацию модели.

Задание 14.4. Провести вычислительный эксперимент с моделью.

Задание 14.5. Сравнить результаты моделирования, полученные в среде Arena и AnyLogic выполнить анализ, полученных результатов.

Содержание отчета и его форма

Подготовьте отчет, в котором приведите технологию выполнения заданий.

Отчет по лабораторной работе должен содержать:

1) название работы;
2) цель лабораторной работы;
3) формулировку задания и технологию его выполнения;
4) ответы на контрольные вопросы;
5) приложение – файлы выполненных заданий.

Вопросы для защиты работы

1. Для чего используется среда имитационного моделирования AnyLogic?
2. Как осуществляется моделирование в среде AnyLogic?
3. На моделирование каких систем ориентирована среда AnyLogic?
4. Какое основное преимущество среды AnyLogic по сравнению с аналогичными системами имитационного моделирования?
5. С помощью среды AnyLogic можно моделировать систем массового обслуживания?

ЛИТЕРАТУРА И ИСТОЧНИКИ

Основная литература

1. Горлач, Б.А., Шахов, В.Г. Математическое моделирование. Построение моделей и численная реализация Санкт-Петербург: Лань, 2021 ЭБС

2. Губарь, Ю.В. Введение в математическое моделирование: учебное пособие Москва: Интернет- Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021 ЭБС

3. Юдович, В.И. Математические модели естественных наук Санкт-Петербург: Лань, 2021ЭБС

Дополнительная литература

1. Безруков Алексей Иосифович, Алексенцева Ольга Николаевна Математическое и имитационное моделирование: Учебное пособие Москва: ООО "Научно- издательский центр ИНФРА-М", 2019 ЭБС

2. Павлова, А.И. Искусственные нейронные сети: учебное пособие Москва: Ай Пи Ар Медиа, 2021 ЭБС

3 Боев, В.Д. Концептуальное проектирование систем в AnyLogic и GPSS World: учебное пособие Москва: Интернет- Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021 ЭБС

4. Афонин, В.В. Моделирование систем Электронный ресурс : учебное пособие / С.А. Федосин / В.В. Афонин. - Моделирование систем,2019-12-01. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 269 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-9963-0352-6

5. Алгазинов, Э.К. Анализ и компьютерное моделирование информационных процессов и систем / Э.К. Алгазинов, А.А. Сирота. – М.: Диалог-МИФИ, 2009. – 416с.

6. Васильев, В.В. Математическое и компьютерное моделирование процессов и систем в среде MATLAB/SIMULINK. Учебное пособие для аспирантов и аспирантов / В.В. Васильев, Л.А. Симак, А.М. Рыбникова. - К.: НАН Украины, 2008. – 91 с.

7.Шагрова Г. В., Романенко М. Г., Топчиев И. Н. Методы исследования и моделирования информационных процессов и технологий: лабораторный практикум. Ставрополь: СКФУ, 2016. - 241 с.: http://biblioclub.ru/.

8. Шагрова, Г. В. Романенко М. Г., Топчиев И. Н. Методы исследования и моделирования информационных процессов и технологий: учебное пособие. Ставрополь: СКФУ, 2016. - 180 с.

9. Богатиков, В.Н. Построение систем управления на основе нейронных сетей: Учебно–методическое пособие / В.Н. Богатиков, Л.В. Дранишников, А.Е. Пророков. - Апатиты: Изд-во КФ ПетрГУ, 2011. – 41 с.

10. Вакуленко С.А., Жихарева А.А. Практический курс по нейронным сетям – СПб: Университет ИТМО, 2018. – 71 с.

8.2. Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине (модулю)

1. Есипов, Б. А. Методы исследования операций: учеб. пособие для вузов / Б. А. Есипов. - 2-е изд., испр. и доп. - Санкт-Петербург; Москва ; Краснодар : Лань, 2013. - 299 с. : ил., табл. ; 21 см. - (Учебники для вузов. Специальная литература). - Гриф: Доп. УМО. - Библиогр.: с. 294-296.

8.3. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля)

1. http://biblioclub.ru – ЭБС «Университетская библиотека on-line», http://old.exponenta.ru/ – Образовательный математический сайт по математике и программированию
2. http://MATLAB.exponenta.ru/Simulink/default.php – сайт, посвященный интерактивному инструменту для моделирования Simulink,
3. https://www.intuit.ru/ – Национальный Открытый Университет «ИНТУИТ»
4. http://window.edu.ru/ – Информационная система "Единое окно доступа к образовательным ресурсам"
5. http://artspb.com/ – Образовательный портал: математика, кибернетика и программирование дополнительного образования
6. http://www.arenasimulation.com – Официальный сайт разработчика среды имитационного моделирования ARENA
7. https://www.anylogic.ru – Официальный сайт разработчика AnyLogic

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Пятигорский институт (филиал) СКФУ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ СТУДЕНТОВ ПО ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ

МОДЕЛИ И МЕТОДЫ ИССЛЕДОВАНИЯ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ И СИСТЕМ

Направление подготовки	**09.04.02 Информационные системы и технологии**
Направленность (профиль)	**«Технологии работы с данными и знаниями, анализ информации»**
Квалификация выпускника	Магистр

Пятигорск, 2024

СОДЕРЖАНИЕ

1. ЦЕЛЬ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Целью освоения дисциплины «Модели и методы исследования информационных процессов и систем» является формирование набора общекультурных компетенций будущего магистра по направлению подготовки 09.04.02 «Информационные системы и технологии», для решения прикладных задач в рамках магистерской программы «Технологии работы с данными и знаниями, анализ информации».

Дисциплина «Методы исследования и моделирования информационных процессов и технологий» должна обеспечивать формирование фундамента подготовки будущих магистров в области телекоммуникаций и интеллектуальных систем, а также, создавать необходимую базу для успешного овладения последующими специальными дисциплинами учебного плана. Она должна способствовать развитию творческих способностей студентов, умению формулировать и решать задачи изучаемой специальности, умению творчески применять и самостоятельно повышать свои знания.

Эти цели достигаются па основе фундаментализации, интенсификации и индивидуализации процесса обучения путём внедрения и эффективного использования современных компьютерных технологий. В результате изучения дисциплины у студентов должны сформироваться знания, умения и навыки, позволяющие проводить самостоятельный анализ инфотелекоммуникационных систем.

Дисциплина является первой дисциплиной, в которой студенты изучают основные методы исследования и моделирование информационных процессов и технологий. Она находится на стыке дисциплин, обеспечивающих базовую и специальную подготовку студентов. Изучая эту дисциплину, студенты впервые знакомятся с современными методами исследования и моделирования информационных процессов и технологий.

Задачи освоения дисциплины: изучение основных методов исследования.

В результате освоения дисциплины обучающийся должен:

Знать:

- основные логические методы и приемы научного исследования, методологические теории и принципы современной науки;
- математический аппарат, описывающий взаимодействие информационных процессов и технологий на информационном, программном и техническом уровнях, теорию нейронных сетей и принципы их использования при проектировании информационных систем;
- историю развития и современные проблемы информатики и вычислительной техники, взаимосвязь и преемственность информационных технологий;
- основные стандарты информационных технологий и информационной безопасности;
- методы разработки компонентов программных комплексов с использованием современных программных средств и технологий разработки алгоритмов и программ, методы отладки;
- основные логические методы и приемы научного исследования, методологические теории и принципы современной науки;
- основные стратегии проектирования современных информационных систем и пути их применения;

- критерии эффективности проектирования информационных систем в различных областях деятельности;
- основные стратегии проектирования современных информационных систем и пути их применения;
- критерии эффективности проектирования информационных систем в различных областях деятельности;
- область применения современных информационных систем;
- методы решения практических задач в области методов исследования и моделирования информационных процессов и технологий;
- методы моделирования процессов, на базе стандартных пакетов автоматизированного проектирования и исследований;
- средства моделирования объектов, на базе стандартных пакетов автоматизированного проектирования и исследований;

Уметь:

- осуществлять методологическое обоснование научного исследования;
- применять современные методы научных исследований для формирования суждений и выводов по проблемам информационных технологий и систем;
- осуществлять математическую постановку исследуемых задач, применять аппарат нейронных сетей в области информационных технологий;
- разрабатывать программные системы с устойчивой архитектурой;
- эффективно использовать на практике теоретические компоненты науки: понятия, суждения, умозаключения, законы;
- осуществлять методологическое обоснование научного исследования;
- применять современные методы научных исследований для формирования суждений и выводов по проблемам информационных технологий и систем;
- разрабатывать информационные системы с применением классических технологий проектирования;
- корректировать структуру современных информационных систем в процессе проектирования при изменении целей и задач проекта;
- воспроизводить знания для практической реализации новшеств.
- проводить разработку и прогнозирование качества процессов функционирования информационных систем и технологий;
- проводить исследование методик анализа, синтеза, оптимизации процессов функционирования информационных систем и технологий;
- осуществлять моделирование процессов на базе стандартных пакетов автоматизированного проектирования и исследований;
- осуществлять моделирование объектов на базе стандартных пакетов автоматизированного проектирования и исследований;

Владеть:

- навыками логико-методологического анализа научного исследования и его результатов; методами научного поиска и интеллектуального анализа научной информации при решении новых задач;
- навыками работы с компьютером, как средством проектирования и отладки ПО и управления информацией;
- навыками работы с основными типами современных информационных систем;
- навыками классификации информации по различным признакам;
- навыками самостоятельной научно-исследовательской работы в частности: методами построения современных проблемно-ориентированных прикладных программных средств;
- навыками логико-методологического анализа научного исследования и его результатов;

- методами научного поиска и интеллектуального анализа научной информации при решении новых задач;
- навыками определения критериев эффективности информационных систем;
- навыками формулировки целей и задач проектирования информационной системы на ранних стадиях ее разработки.
- навыками проведения разработки и исследования теоретических и экспериментальных моделей объектов профессиональной деятельности в областях.
- навыками разработки и прогнозирования качества процессов функционирования информационных систем и технологий;
- навыками исследования методик анализа, синтеза, оптимизации процессов функционирования информационных систем и технологий;
- навыками моделирование процессов на базе стандартных пакетов автоматизированного проектирования и исследований;
- навыками моделирование объектов на базе стандартных пакетов автоматизированного проектирования и исследований;

2. ТЕХНОЛОГИЧЕСКАЯ КАРТА САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТА

Для студентов очной формы обучения:

Коды реализуемых компетенций, индикатора(ов)	Вид деятельности студентов	Средства и технологии оценки	Объём часов, в том числе		
			СРС	Контактная работа с преподавателем	Всего
2 семестр					
ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2	Подготовка к лекциям	Коллоквиум	1,62	0,18	1,8
ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2	Подготовка к лабораторной работе	Отчет письменный	4,86	0,54	5,4
ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2	Самостоятельное изучение литературы	Коллоквиум	74,52	8,28	82,8
Итого за 2 семестр			81	9	90
Итого			81	9	90

Для студентов заочной формы обучения:

Коды реализуемых компетенций, индикатора(ов)	Вид деятельности студентов	Средства и технологии оценки	Объём часов, в том числе		
			СРС	Контактная работа с преподавателем	Всего
1 семестр					
ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2	Подготовка к лекциям	Коллоквиум	0,36	0,04	0,4
ОПК-7.1 ОПК-7.2 ОПК-8.1	Подготовка к лабораторной	Отчет письменный	140,58	15,62	156,2

ОПК-8.2	работе				
ОПК-7.1 ОПК-7.2 ОПК-8.1 ОПК-8.2	Самостоятельное изучение литературы	Коллоквиум	2,16	0,24	2,4
Итого за 1 семестр			**143,1**	**15,9**	**159**
Итого			**143,1**	**15,9**	**159**

3. Организационно-методические рекомендации по освоению дисциплины

4.

Самостоятельная работа студентов является важнейшим условием формирования научного способа познания. Она проводится накануне

каждого лабораторного занятия и включает изучение необходимого для выполнения лабораторной работы теоретического материала, а также подготовку отчета по выполненным на лабораторном занятии заданиям.

Подготовленный материал оформляется в виде отчета.

Самостоятельные занятия (СЗ) являются одной из активных форм обучения.

Самостоятельные занятия по дисциплине «Методы исследования и моделирования информационных процессов и технологий» имеют целью:

- закрепить и углубить знания, полученные студентами на лекциях и в процессе лабораторных занятий;
- привить практические навыки при разработке, эксплуатации и исследовании информационных процессов, систем и тебхнологий.

Самостоятельные занятия проводятся в специализированных аудиториях, оборудованных СВТ и возможностью пользования Интернет-ресурсами, в библиотеке, а также дома.

Предлагаемые методические рекомендации содержат информацию для студентов, необходимую при подготовке и проведении лабораторных занятий по дисциплине «Методы исследования и моделирования информационных процессов и технологий».

5. Методические указания по выполнению самостоятельной работы студентов

Методические указания должны включать следующие разделы:

- цель работы;
- задание, которое должно быть выполнено студентом в результате
- проведения самостоятельной работы;
- варианты индивидуальных заданий;
- основные теоретические положения, необходимые для
- выполнения задания, они должны быть краткими и содержать ссылки на
- литературу, в которой эти положения изложены в объеме, достаточном для
- выполнения самостоятельной работы;
- этапы выполнения задания с указанием конкретных сроков

- выполнения каждого из этапов и всего задания в целом;
- требования к оформлению графической и текстовой части
- самостоятельной работы;
- пример выполнения одного из вариантов задания и оформления
- отчета;
- библиографический список использованных источников.
-

6. СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Наименование разделов и тем дисциплины, их краткое содержание; вид самостоятельной работы представлено в таблице 2.

Таблица 2 – Наименование разделов и тем дисциплины, их краткое содержание; вид самостоятельной работы

№	Наименование разделов и тем дисциплины, их краткое содержание; вид самостоятельной работы	Форма контроля
	1 семестр	
1	подготовка к лабораторным занятиям	защита
	Самостоятельное изучение тем: - характеристика и модели базовых информационных процессов. - задачи анализа и синтеза систем. - моделирование и проектирование сложных систем. - моделирование динамических систем.	собеседование
	выполнение контрольной работы	Защита
	2 семестр	
	подготовка к лабораторным занятиям	защита
	Самостоятельное изучение тем: - характеристика и модели базовых - информационных процессов. - задачи анализа и синтеза систем. - моделирование и проектирование сложных систем. - моделирование динамических систем.	собеседование
	выполнение курсовой работы	Защита

Наименование тем дисциплины, цель, форма контроля, задания, требования к оформлению , перечень литературных источников представлен в таблице3.

Таблица 3 – Наименование тем дисциплины, цель, форма контроля, задания и требования к оформлению самостоятельной работы

Название темы	Цель	Форма контроля	Задания для СРС	Требования
Подготовка к лекциям и лабораторным занятиям	изучить процессы и поколения управления данными, а также тенденции их развития	индивидуальное собеседование	Сформулировать ответы на контрольные вопросы к лабораторным работам	Устно ответить на контрольные вопросы к лабораторным работам

Характеристики и модели базовых информационных процессов.	изучить модели базовых информационных процессов.	индивидуальное собеседование	сформулировать и письменно ответить на вопросы для контроля владения компетенциями данного раздела программы	письменно ответить на вопросы для контроля владения компетенциями вопросы данного раздела программы
Задачи анализа и синтеза систем.	Изучить задачи анализа и синтеза систем.			
Моделирование и проектирования сложных систем.	изучить методологию моделирования и проектирования сложных систем.			
. Морфологические методы; типовые	изучить морфологические методы			
Математические схемы элементов сложной системы	изучить математические схемы элементов сложной системы			

7. Примерная тематика заданий для самостоятельной работы студентов

Учебным планом предусмотрено выполнение курсовой работы (сроки выполнения 2-й семестр, форма контроля – защита)

Примерные темы курсовых работ по дисциплине:

1. Компьютерное моделирование информационных процессов.

Сравнение различных алгоритмов генерации случайного процесса .

2. Построение алгоритмов, основанных на классических методах моделирования случайных процессов, с учетом современных возможностей среды Matlab.
3. Нейросетевые алгоритмы моделирования цветных изображений пространственно распределенных и локализованных объектов.
4. Компьютерное моделирование помех в задачах компьютерной обработки изображений.
5. Исследование и моделирование латентных изображений средствами Matlab.
6. Моделирование каналов передачи информации в Matlab.
7. Разработка компьютерных имитационных моделей систем массового обслуживания средствами Matlab.

8. Моделирование систем массового обслуживания с использованием гибридных автоматов средствами Matlab.

9. Разработка системы компьютерного моделирования динамики намагничивающихся капель в среде Matlab.

10. Моделирование систем с адаптивной структурой.

11 Автоматизированные технологии создания моделей в среде Matlab .

Курсовая работа оформляется в соответствии с требованиями ГОСТов (ГОСТ 2.105-95 ЕСКД. Общие требования к текстовым документам и ГОСТ 2.105-96 ЕСКД. Текстовые документы) и сдается преподавателю в печатном виде (формат А4) и электронном виде: файлы с текстом курсовой работы, программной реализацией. Курсовая работа сдается на проверку в срок не менее чем за неделю до защиты. После проверки студент либо допускается к защите, либо дорабатывает и вносит исправления в соответствии с замечаниями преподавателя. Во время защиты работы студент делает доклад (около 5 минут), в котором кратко излагает результаты ее выполнения.

Доклад должен быть представлен в виде презентации. Примерные темы для самостоятельного изучения по дисциплине «Методы исследования и моделирования информационных процессов и технологий»:

1. Методы анализа качества информационных процессов и систем.
2. Современные средства вычислительной техники, коммуникаций и связи.
3. Применение методологии системного анализа и информационных технологий при проектировании и исследовании информационных систем.
4. Перспективы и тенденции развития информационных технологий;
5. Порядок, методы и средства защиты интеллектуальной
6. собственности.
7. Основные требования к организации труда при исследовании и разработке информационных процессов и систем.
8.

8. План-график выполнения СРС

№	Название раздела	Срок сдачи результатов
1	Методология анализа и моделирования информационных процессов и систем	5
2	Применение методов системного анализа и информационно-аналитических технологий при проектировании информационных систем	10
3	Компьютерное моделирование информационных процессов и технологий	15

9. Организация контроля знаний студентов

8.1. Формы контроля знаний студентов

Контроль и оценка знаний, умений и навыков студентов осуществляется на лабораторных занятиях, консультациях, при сдаче

экзамена. В ходе контроля знаний преподаватель оценивает понимание студентом содержания дисциплины «Методы исследования и моделирования информационных процессов и технологий», его способность анализировать развитие информационных систем и технологий.

Контроль знаний студентов может осуществляться в следующих

формах:

- текущий контроль знаний;
- итоговый контроль знаний.

Текущий контроль знаний студентов имеет целью: дать оценку работы каждого студента по усвоению им учебного материала, выявить недостатки в его подготовке и оказать практическую помощь в их устранении;

Основными формами текущего контроля знаний студентов являются:

- устный контрольный опрос;
- защита лабораторной работы;
- проверка конспектов лекций.

Устный контрольный опрос студентов проводится на лекциях (и лабораторных занятиях). По его результатам преподаватель оценивает качество подготовки студента к занятию.

На лабораторных занятиях знания и практические навыки студентов оцениваются по 4-балльной системе. Полученные оценки выставляется в журнале.

При проверке конспектов лекций дается анализ качества их ведения. Отмечаются допущенные ошибки, в рецензии преподавателя оценивается качество конспектирования учебного материала, даются рекомендации по улучшению качества конспектирования лекционного материала.

8.2. Рекомендации по подготовке к экзамену

Подготовка к зачету начинается с начала изучения дисциплины.

Необходимо посещать все лекции и лабораторные занятия. Экзамен, как итоговый контроль знаний студентов имеет целью проверить и оценить учебную работу студентов, уровень полученных знаний и практических навыков. Экзамен проводится в 1 семестре после защиты всех лабораторных работ в объеме учебной программы.

10. Рекомендации по работе с литературой и источниками

Изучение литературы и источников необходимо начинать с прочтения соответствующих глав учебных изданий, учебных пособий или литературы, рекомендованной в качестве основной или дополнительной по дисциплине «Методы исследования и моделирования информационных процессов и технологий», которые прямо или косвенно относятся к изучаемой теме.

При изучении литературы и источников студенту рекомендуется вести краткий конспект. Однако не следует переписывать все содержание изучаемой темы, нужно выписывать лишь основные идеи и главные на ваш взгляд мысли. В отдельных случаях, когда встречаются важные определения, понятия, необходимый фактический материал и примеры, статистическая информация, имеющие отношение к изучаемой теме, студенту следует выписать их в виде цитат с полным указанием библиографических источников.

Конспектирование рекомендуемой литературы и источников необходимо вести с распределением собранных материалов по отдельным главам и параграфам согласно учебно-тематическому плану. Необходимо выписывать все выходные данные по используемой литературе и источникам.

Основой технологии интенсификации обучения на платформе

цифровых образовательных технологий являются учебно-иллюстрационные материалы (опорный конспект) по дисциплине «Методы исследования и моделирования информационных процессов и технологий».

Работа с учебно-иллюстрационными материалами имеет следующие этапы.

1. Изучение теоретических основ учебного материала в аудитории: изложение преподавателем изучаемого материала студентам с объяснением по опорному конспекту;

2. Самостоятельная работа: индивидуальная работа студентов по опорному конспекту; фронтальное закрепление по блокам опорного конспекта.

3. Первое повторение - воспроизведение содержания заданной темы опорного конспекта по памяти.

4. Устное проговаривание материала опорного конспекта–

необходимый этап внешнеречевой деятельности при усвоении учебного материала.

5. Второе повторение – взаимоопрос и взаимопомощь студентов друг другу.

Применение учебно-иллюстрационных материалов позволяет

обобщить сложный по содержанию материал, активизировать мыслительную деятельность студентов.

Необходимо помнить, что главное для студента в самостоятельной работе с рекомендуемой литературой и источниками - это формирование своего индивидуального стиля, который может стать основой в будущей профессиональной деятельности.

11. КРИТЕРИИ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ

Оценка «отлично» выставляется студенту, если глубокие, исчерпывающие знания и творческие способности в понимании, изложении и использовании учебно-программного материала; логически последовательные, содержательные, полные, правильные и конкретные ответы на все поставленные вопросы и дополнительные вопросы преподавателя; свободное владение основной и дополнительной литературой, рекомендованной учебной программой.

Оценка «хорошо» выставляется студенту, если твердые и достаточно полные знания всего программного материала, правильное понимание сущности и взаимосвязи рассматриваемых процессов и явлений; последовательные, правильные, конкретные ответы на поставленные вопросы при свободном устранении замечаний по отдельным вопросам; достаточное владение литературой, рекомендованной учебной программой.

Оценка «удовлетворительно» выставляется студенту, если твердые знания и понимание основного программного материала; правильные, без грубых ошибок ответы на поставленные вопросы при устранении неточностей и несущественных

ошибок в освещении отдельных положений при наводящих вопросах преподавателя; недостаточное владение литературой, рекомендованной учебной программой.
Оценка «неудовлетворительно» выставляется студенту, если неправильные ответы на основные вопросы, допущены грубые ошибки в ответах, непонимание сущности излагаемых вопросов; неуверенные и неточные ответы на дополнительные вопросы.

12. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

11.1. Рекомендуемая литература

11.1.1. Основная литература:

1. Советов, Борис Яковлевич Моделирование систем: для студентов высших учебных заведений, обучающихся по направлениям "Информатика и вычислительная техника" и "Информационные системы" / Б.Я. Советов, С.А. Яковлев; Санкт-Петербургский гос. электротехнический ун-т. – Москва Юрайт, 2012. – 342 с.
2. Учебное пособие по дисциплине «Автоматизированное проектирование средств и систем управления» Пятигорск 2012. Под редакцией проф. Першина И.М.

11.1.2. Дополнительная литература:

1. Вдовин В.М., Суркова Л.Е, Валентинов В.А. Теория систем и системный анализ – М.: ИТК «Дашков и К°», 2010. – 320 с Прохоров Н.Л. (и др.) Управляющие вычислительные комплексы: учеб. пособие / ред. Н.Л. Прохоров, 2010 г. – 352 с.
2. Андрейчинков А.В. Интеллектуальные информационные системы: учебник / А.В. Андрейчинков, О.Н. Анрейчикова, 2010 г. – 424 с
3. Благодатских В.А. Стандартизация разработки программных средств: учеб. пособие / В.А. Благодатских, В.А. Волнин, К.Ф. Поскакалов; под ред. Проф. О.С. Разумова, 2010 г. - 288 с.

11.1.3. Методическая литература:

1. Методические указания по выполнению лабораторных работ по дисциплине «Методы исследования и моделирования информационных процессов и технологий».
2. Методические рекомендации для студентов по организации самостоятельной работы по дисциплине «Методы исследования и моделирования информационных процессов и технологий».
3. Методические указания по выполнению курсовой работы по дисциплине «Методы исследования и моделирования информационных процессов и технологий».

11.1.4. Интернет-ресурсы:

1. http://www.biblioclub.ru - электронная библиотечная система «Университетская библиотека – online»: специализируется на учебных материалах для ВУЗов по научно-гуманитарной тематике, а так же содержит материалы по точным и естественным наукам.

2. http://www.iprbookshop.ru– электронно-библиотечная система IPRbooks.