МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ДГТУ)

Кафедра «Математики и информатики»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
для проведения лабораторных занятий по дисциплине «Модели и методы исследования информационных процессов и систем»

Ростов-на-Дону
ДГТУ
2022

УДК 004.8(075.8)

Составитель: А.И. Сухинов

Методические указания для проведения лабораторных занятий по дисциплине «Модели и методы исследования информационных процессов и систем» – Ростов-на-Дону : Донской гос. техн. ун-т, 2022. – 33 с.

Методические указания содержат задания к лабораторным работам по освоению математических моделей и методов моделирования, применяемых в сфере информационных технологий и используемых при разработке информационных систем. Описываются и характеризуются различные формальные модели систем, как дискретных, так и стохастических.

Предназначены для обучающихся по направлению подготовки магистратуры 09.04.02 «Информационные системы и технологии».

УДК 004.8(075.8)

Печатается по решению редакционно-издательского совета
Донского государственного технического университета

Ответственный за выпуск зав. кафедрой «Математика и информатика» д-р физ.-мат. наук, профессор А.И. Сухинов

_____

В печать _____ г.
Формат 60×84/16. Объем _____усл. п. л.
Тираж _____экз. Заказ № _____

_____

Издательский центр ДГТУ
Адрес университета и полиграфического предприятия:
344000, г. Ростов-на-Дону, пл. Гагарина, 1

Лабораторная работа 1. Имитационное моделирование систем массового обслуживания

Цель работы: научиться использовать язык GPSS (General Purpose Simulation System) для исследования процедур имитационного моделирования сложных технических объектов, представленных как системы массового обслуживания.

Язык GPSS (General Purpose Simulation System), ориентированный на процессы, разработан еще в 1961 г., но продолжает широко использоваться. Язык реализован в ряде программ имитационного моделирования, так, версия программы GPSS/PC в среде Windows создана в 2000 г.

Модель (программа) на языке GPSS представляет собой последовательность операторов (их называют блоками), отображающих события, происходящие в СМО при перемещениях транзактов. Поскольку в интерпретаторах GPSS реализуется событийный метод и в СМО может быть одновременно много транзактов, то интерпретатор будет попеременно исполнять разные фрагменты программы, имитируя продвижения транзактов в текущий момент времени до их задержки в некоторых устройствах или очередях.

Операторы (блоки) GPSS имеют следующий формат:

<метка> <имя_оператора> <поле_операндов> [<комментарий>]

Метка может занимать позиции, начиная со второй, имя оператора — с восьмой, поле операндов — с девятнадцатой, комментарий обязательно отделяется от поля операндов пробелом.

Поле операндов может быть пусто, иметь один или более операндов, обозначаемых ниже при описании блоков символами **A**, **B**, **C**,... Операндами могут быть идентификаторы устройств, накопителей, служебные слова и *стандартные числовые атрибуты* (СЧА). К СЧА относятся величины, часто встречающиеся в разных задачах. Это, например, такие операнды, как **S** — объем занятой памяти в накопителе, **F** — состояние устройства, **Q** —

текущая длина очереди, **P** — параметр транзакта (каждый транзакт может иметь не более **L** параметров, где **L** зависит от интерпретатора), **V** — целочисленная переменная (вещественная и булева переменные обозначаются **FV** и **BV** соответственно), **X** — хранимая переменная (переменная, для которой автоматически подсчитывается статистика), **K** — константа, **AC1** — текущее время, **FN** — функция, **RN** — случайная величина, **RN1** — случайная величина, равномерно распределенная в диапазоне [0, 1] и др. При этом ссылки на СЧА записываются в виде **<СЧА>$<идентификатор>**. Например, **Q$ORD** означает очередь ORD или **FN$COS** — ссылка на функцию COS.

Рассмотрим наиболее часто встречающиеся операторы, сопровождая знакомство с ними простыми примерами моделей.

Источники заявок обычно описываются блоком

GENERATE A,B,C,D,E

Здесь **A** и **B** служат для задания интервалов между появлениями заявок, при этом можно использовать один из следующих вариантов:

- интервал — равномерно распределенная в диапазоне [A_B, A+B] случайная величина;
- интервал — значение функции, указанной в B, умноженной на A;

C — задержка в выработке первого транзакта; **D** — число вырабатываемых источником заявок; **E** — приоритет заявок. Если **D** пусто, то число вырабатываемых транзактов неограничено. Например:

GENERATE 6,FN$EXP,,15

Этот оператор описывает источник, который вырабатывает 15 транзактов с интервалами, равными произведению числа 6 и значения функции EXP;

GENERATE 36,12

Здесь число транзактов неограничено, интервалы между транзактами — случайные числа в диапазоне [24, 48].

Функции, на которые имеются ссылки в операторах, должны быть описаны с помощью блока следующего типа:

M FUNCTION A,B

За ним следует строка, начинающаяся с первой позиции :

X1,Y1/X2,Y2/X3,/.../Xn,Yn

Здесь метка **M** — идентификатор функции, **A** — аргумент функции, **B** — тип функции, **Xi** и **Yi** — координаты узловых точек функции, заданной таблично.

Например:

EXP FUNCTION RN1,C12

0,0/.2,.22/.4,.51/.5,.6/.6,.92/.7,1.2/.8,1.61/.9,2.3/.95,3/.99,4.6/.999,6.9/1,1000

Это описание непрерывной (**C**) функции EXP, заданной таблично 12-ю узловыми точками, аргументом является случайная величина (RN1), равномерно распределенная в диапазоне [0, 1]; или:

BBB FUNCTION *4,D6

1,2/2,5/3,11/4,20/5,18/6,12/7,9

Дискретная (**D**) функция BBB задана 6-ю узловыми точками, аргумент — четвертый параметр транзакта, возбудившего обращение к функции BBB.

Здесь аргумент задан с использованием косвенной адресации, признаком которой является символ *, т.е. запись *4 означает, что аргументом является величина, указанная в 4-м параметре транзакта, вызвавшего функцию (в данном примере можно было бы использовать равноценную запись *p4). В общем случае косвенная адресация выполняется путем записи операнда в виде СЧА*СЧА, например: S*X2, что означает емкость памяти, занятая в накопителе, именем которого является значение переменной X2, или Q*p5 — длина очереди с именем, записанным в парамтере 5 транзакта.

Тразакты могут порождаться и оператором размножения:

SPLIT A,B,C

Новые транзакты порождаются, когда в данный блок входит некоторый транзакт. При этом создается семейство транзактов, включающее основной (вошедший в блок) транзакт и A его копий. Основной транзакт переходит в следующий по порядку блок, а его копии переходят в блок с меткой B. Для различения транзактов параметр C основного транзакта увеличивается на 1, а транзактов-копий — на 2, 3, 4,... и т.д.

Обратное действие — сборка транзактов выполняется операторами:

ASSEMBLE A

GATHER A

Согласно оператору ASSEMBLE первый из вошедших в блок транзактов выйдет из него только после того, как в этот блок придут еще A-1 транзактов того же семейства. Второй оператор отличается от предыдущего тем, что из блока выходят все A транзактов.

Операторы занятия транзактом и освобождения от обслуживания устройства A:

SEIZE A

RELEASE A

Задержка в движении транзакта по СМО описывается оператором:

ADVANCE A,B

A и B имеют тот же смысл, что и в операторе **GENERATE**.

Пример 1

Обслуживание транзакта в устройстве WST продолжительностью a единиц времени, где a — равномерно распределенная в диапазоне [7,11] случайная величина, описывается следующим фрагментом программы

SEIZE WST

ADVANCE 9,2

RELEASE WST

Аналогично описывается занятие транзактом памяти в накопителе:

ENTER A,B

Здесь помимо имени накопителя (A) указывается объем занимаемой памяти (B).

Освобождение B ячеек памяти в накопителе A выполняется оператором:

LEAVE A,B

Для накопителей в модели нужно задавать общий объем памяти, что делается в следующем описании накопителя:

M STORAGE A

Здесь: M — имя накопителя, A — объем его памяти.

Если транзакт приходит на вход занятого устройства или на вход накопителя с недостаточным объемом свободной памяти, то транзакт задерживается в очереди к этому устройству или накопителю. Слежение за состоянием устройств и очередей выполняет интерпретатор. Но если в модели требуется ссылаться на длину очереди или собирать статистику по ее длине, то требуется явное указание этой очереди в модели. Делается это с помощью операторов входа в очередь и выхода из очереди:

QUEUE A,B

DEPART A,B

Согласно этим операторам, очередь A увеличивается и уменьшается на B единиц соответственно, если B=1, то поле B можно оставить пустым.

Движение транзактов выполняется по маршруту, заданному последовательностью операторов в модели. Если требуется изменение естественного порядка, то используется оператор перехода. Оператор условного перехода имеет вид:

TEST XX A,B,C

В соответствии с ним переход к оператору, помеченному меткой C,

происходит, если не выполняется условие A XX B, где XX ∈ {**E**, **NE**, **L**, **LE**, **G**, **GE**}; **E** — равно; **NE** — неравно; **L** — меньше; **LE** — меньше или равно; **G** — больше; **GE** — больше или равно (XX всегда размещается в позициях 13 и 14).

Пример 2

Приходящие пользователи ожидают обслуживания, если длина очереди не более 4, иначе от обслуживания отказываются. Соответствующий фрагмент программы

```
TEST LE    Q$STR,4,LBL
QUEUE      STR
SEIZE      POINT
DEPART     STR
ADVANCE    50,16
RELEASE    POINT
...
LBL   TERMINATE  1
```

В примере 2 использован оператор выхода транзактов из СМО:

```
TERMINATE  A
```

Согласно этому оператору из итогового счетчика вычитается число A. С помощью итогового счетчика задается длительность моделирования. В начале исполнения программы в счетчик заносится число, указанное в операнде A оператора

```
START      A,B,C
```

Моделирование прекращается, когда содержимое счетчика будет равно или меньше нуля. Операнд C — шаг вывода статистики на печать. Если B=0 и C=0, то выполняется только стандартная печать по окончании моделирования. В

стандартную печать входят собранные за время моделирования статистические данные по основным параметрам модели: средние и максимальные значения длин очередей, объемов занимаемой памяти в накопителях, времени занятого состояния устройств и др. От печати можно отказаться, указав B=NP.

Пример 3

Общая структура программы на GPSS имеет вид

SIMULATE

<описания, в том числе функций, накопителей, массивов и т.п.>

<операторы, моделирующие движение транзактов>

START A,B,C

END.

Оператор безусловного перехода записывается следующим образом:

TRANSFER ,B

Здесь B — метка оператора, к которому следует переход.

Используется ряд других разновидностей оператора **TRANSFER**. Например:

TRANSFER P,B,C

Переход происходит к оператору с меткой, равной сумме значения параметра B транзакта и числа C.

TRANSFER FN,B,C

То же, но вместо параметра транзакта слагаемым является значение функции B.

TRANSFER PICK,B,C

Это оператор равновероятного перехода к операторам, метки которых находятся в интервале [B,C].

Важное место в СМО занимает переход по вероятности:

TRANSFER A,B,C

Здесь A — вероятность перехода к оператору с меткой C, переход к оператору с меткой B будет происходить с вероятностью 1 – A.

Оператор перехода в циклических процедурах выглядит так:

LOOP A,B

Здесь A — номер параметра транзакта, в котором содержится число повторений (витков) цикла, B — метка оператора, с которого начинается повторяющаяся часть.

Пример 4

Заказы, поступающие в СМО в случайные моменты времени в диапазоне [20,40], выполняет сначала бригада WGR1, затем параллельно работают бригады WGR2 и WGR3, каждая над своей частью заказа. Заданы экспоненциальные законы для времен выполнения работ бригадами WGR1, WGR2 и WGR3 с интенсивностями 0,05, 0,1 и 0,125 соответственно. Моделирование нужно выполнить на временном отрезке, соответствующем выполнению 1000 заказов.

Программа:

```
SIMULATE
EXP FUNCTION RN1,C12
0,0/.2,.22/.4,.51/.5,.6/.6,.92/.7,1.2/.8,1.61/.9,2.3/.95,3/.99,4.6/.999,6.9/1,1000
GENERATE 30,10
SEIZE WGR1
ADVANCE 20,FN$EXP
RELEASE WGR1
SPLIT 1,MET1
SEIZE WGR2
ADVANCE 10,FN$EXP
```

```
      RELEASE    WGR2
      TRANSFER   ,MET2
MET1  SEIZE      WGR3
      ADVANCE    8,FN$EXP
      RELEASE    WGR3
MET2  ASSEMBLE   2
      TERMINATE  1
      START      1000
      END.
```

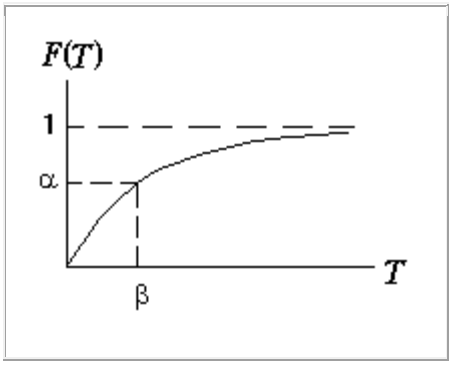


Рисунок 1- Функция экспоненциального закона распределения

В этом примере использован экспоненциальный закон распределения с плотностью

$$P(t) = \lambda \exp(-\lambda t),$$

где λ — интенсивность. Функция распределения экспоненциального закона

$$F(T) = \int_0^T p(t)\,dt = 1 - \exp(-\lambda T).$$

Из рис. 1 ясно, что поскольку искомыми являются значения β случайной величины T, то, задавая значение α, как равномерно распределенной в диапазоне $[0, 1]$ случайной величины, по формуле

$$\beta = \frac{1}{\lambda} \ln \frac{1}{1-\alpha} \qquad (1)$$

находим искомое значение. Именно в соответствии с (1) в операторах **ADVANCE** множителями были значения $\frac{1}{\lambda}$.

Приведем еще несколько операторов языка GPSS.

Оператор изменения параметров транзактов:

ASSIGN A,B

Здесь A — номер параметра транзакта, B — присваиваемое ему значение.

В следующих операторах параметр A увеличивается (уменьшается) на значение B:

ASSIGN A+,B

ASSIGN A-,B

Оператор фиксации времени события

MARK A

его выполнение заключается в запоминании значения текущего модельного времени в параметре вошедшего в оператор MARK транзакта, в поле A указывается номер параметра.

Пример 5

На вход производственной линии поступают и проходят обработку на станке TOOL1 детали типов X и Y. Далее детали типа X обрабатываются на станке TOOL2, а детали типа Y — на станке TOOL3. Интервал моделирования соответствует обработке 600 деталей (ниже у операторов **GENERATE** и **ADVANCE** значения операндов не конкретизированы):

```
SIMULATE
GENERATE   A,B
ASSIGN     1,LBL4
TRANSFER   ,LBL1
GENERATE   A,B
```

```
    ASSIGN    1,LBL2
LBL1 SEIZE    TOOL1
    ADVANCE   A,B
    RELEASE   TOOL1
    TRANSFER  P,1
LBL4 SEIZE    TOOL2
    ADVANCE   A,B
    RELEASE   TOOL2
LBL3 TERMINATE 1
LBL2 SEIZE    TOOL3
    ADVANCE   A,B
    RELEASE   TOOL3
    TRANSFER  ,LBL3
    START     600
    END.
```

Расширение возможностей управления движением транзактов достигается благодаря операторам, реализующим механизм семафора:

```
    LOGIC X   A
    GATE XX   A,B
```

Первый оператор при X = S устанавливает переключатель A в единичное состояние, при X = R сбрасывает его в нулевое состояние, при X = I инвертирует значение состояния. Второй оператор при XX = LR и значнии переключателя, указанного в A, равном 1, или при XX = LS и состоянии переключателя 0 передает транзакт оператору с меткой B (или задерживает его в блоке **GATE**, если поле B пусто), а при других сочетаниях XX и значений переключателя — направляет к следующему оператору.

Вычислительный оператор присваивает переменной с номером M значение арифметического выражения A:

M VARIABLE A

Например в следующем операторе переменной номер 3 присваивается разность числа 216 и объема занятой памяти в накопителе MEM2:

XINIT VARIABLE K216-S$MEM2

Знаки арифметических операций сложения, вычитания, умножения, деления +, -, #, / соответственно. В случае логических выражений имя оператора должно быть BVARIABLE, а знаками операций дизъюнкции и конъюнкции являются + и #. Если операции выполняются над числами типа real, то имя оператора FVARIABLE.

Следующий оператор присваивает хранимой переменной, указанной в A. значение, записанное в B:

SAVEVALUE A,B

Прерывание обслуживания заявки в устройстве A происходит при входе некоторой другой заявки в блок:

PREEMT A

а возобновление прерванного обслуживания заявки — при входе в блок:

RETURN A

Оператор синхронизации, реализующий механизм рандеву, имеет, например, вид:

LBL MATCH NUMB

Приходящий в него транзакт задерживается до тех пор, пока в некоторой другой части модели в сопряженный оператор не войдет транзакт того же семейства. Сопряженный оператор выглядит так:

NUMB MATCH LBL

Часто сведения о некоторых величинах, характеризующих моделируемый

процесс, удобно представлять в виде гистограмм. Задание гистограммы выполняют в разделе описаний с помощью оператора:

M TABLE A,B,C,D

Здесь M — имя гистограммы; A — табулируемая величина; B — верхняя граница левого интервала гистограммы; C — ширина интервалов; D — число интервалов. Формирование гистограммы происходит с помощью оператора:

TABULATE A

Выполнение этого оператора увеличивает на единицу число попаданий в i-й интервал гистограммы, имя которой указано в A. При этом i-й интервал соответствует текущему значению переменной, являющейся аргументом для гистограммы.

Пример 6

Требуется разработать модель процессов возникновения и устранения неисправностей в некоторой технической системе, состоящей из множества однотипных блоков; в запасе имеется один исправный блок; известны статистические данные об интенсивностях возникновения отказов и длительностях таких операций, как поиск неисправностей, замена и ремонт отказавшего блока. Поиск и замену отказавшего блока производит бригада TEAM1, а ремонт замененного блока — бригада TEAM2.

```
SIMULATE
GENERATE   A,B   моделируется возникновение отказов
SEIZE      TEAM1
ADVANCE    A,B   поиск неисправности
ENTER      MEM,1 получение запасного блока из резерва
ADVANCE    A,B   замена блока
RELEASE    TEAM1
SEIZE      TEAM2
```

```
    ADVANCE    A,B   ремонт
    LEAVE      MEM,1 восстановление резерва
    RELEASE    TEAM2
    TERMINATE  1
    START      1000
    END.
```

Пример 7

Требуется разработать модель сборки изделия из 30 деталей типа А1 и 16 деталей типа А2, поступающих на сборочный участок от независимых экспоненциальных источников с интенсивностями λ , равными 0,1 и 0,04 мин$^{-1}$ соответственно. Длительность сборочной операции находится в пределах [12,18] мин. Промоделировать выпуск 600 изделий. Табулировать наполнение входного бункера с деталями типа А2 перед началом сборки.

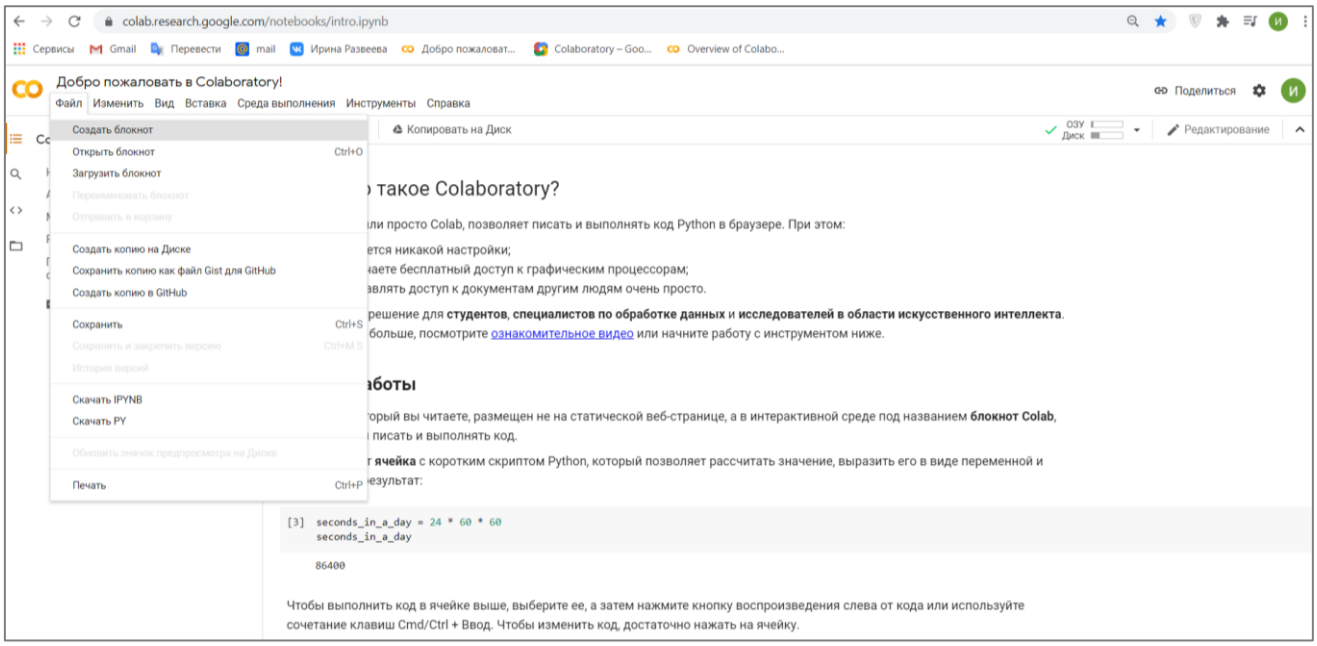
```
    SIMULATE
MEM1  STORAGE     30
MEM2  STORAGE     16
TAB   TABLE       MEM2,32,16,6
EXP   FUNCTION    RN1,C12
0,0/.2,.22/.4,.51/.5,.6/.6,.92/.7,1.2/.8,1.61/
.9,2.3/.95,3/.99,4.6/.999,6.9/1,1000
    GENERATE    10, FN$EXP
    ENTER       MEM1,1
    TRANSFER    ,MMM
    GENERATE    25, FN$EXP
    ENTER       MEM2,1
MMM   TEST GE     S$MEM1,30,LLL
```

```
    TEST GE      S$MEM2,16,LLL
    TABULATE     TAB
    SEIZE        MONT
    ADVANCE      15,3
    RELEASE      MONT
    TERMINATE    1
LLL TERMINATE
    START        600
    END.
```

Лабораторная работа 1. Моделирование нейронных сетей на языке Python с применением пакета Keras

Цель работы: построить нейронную сеть на Python в Google Colaboratory для распознавания предметы одежды из набора данных Fashion MNIST, научиться оценивать влияние гиперпараметров обучения (количество эпох обучения, размер мини-выборки, количество нейронов во входном слое, количество скрытых слоев) на качество обучения нейронной сети.

1. Откройте платформу по ссылке:
https://colab.research.google.com/notebooks/intro.ipynb
2. Перейдя во вкладку Файл выбрать «Создать блокнот».



3. Перед нами пустая рабочая область.

Документ, который вы читаете, размещен не на статической веб-странице, а в интерактивной среде под названием блокнот Colab, позволяющей писать и выполнять код.

В лабораторной работе вы научитесь обучать нейронную сеть распознавать предметы одежды из набора данных Fashion MNIST.

Датасет Fashion MNIST - содержит 70,000 монохромных изображений в 10 категориях. На каждом изображении содержится по одному предмету одежды в низком разрешении (28 на 28 пикселей).

Fashion MNIST предназначен для замены классического датасета MNIST который часто используют как "Hello, World" программ машинного обучения для компьютерного зрения.

Необходимое программное обеспечение:

- библиотека TensorFlow;
- интерфейс для программирования нейросетей Keras.

Код программы:

```
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras import utils
import numpy as np

# Загружаем данные
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

# Список с названиями классов
classes = ['футболка', 'брюки', 'свитер', 'платье', 'пальто',
           'туфли', 'рубашка', 'кроссовки', 'сумка', 'ботинки']

# Преобразование размерности изображений
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
# Нормализация данных
x_train = x_train / 255
x_test = x_test / 255

# Преобразуем метки в категории
```

```
y_train = utils.to_categorical(y_train, 10)
y_test = utils.to_categorical(y_test, 10)

# Создаем последовательную модель
model = Sequential()

# Добавляем уровни сети
model.add(Dense(800, input_dim=784, activation="relu"))
model.add(Dense(10, activation="softmax"))

# Компилируем модель
model.compile(loss="categorical_crossentropy",
              optimizer="SGD",
              metrics=["accuracy"])

print(model.summary())

# Обучаем сеть
history = model.fit(x_train, y_train,
                    batch_size=200,
                    epochs=100,
                    validation_split=0.2,
                    verbose=1)

# Оцениваем качество обучения сети на тестовых данных
scores = model.evaluate(x_test, y_test, verbose=1)
print("Доля верных ответов на тестовых данных, в процентах:", round(scores
[1] * 100, 4))
```

```
240/240 [==============================] - 4s 16ms/step - loss: 0.5071 - accuracy: 0.8328 - val_loss: 0.5050 - val_accuracy: 0.8278
Epoch 10/100
240/240 [==============================] - 4s 17ms/step - loss: 0.4959 - accuracy: 0.8354 - val_loss: 0.4973 - val_accuracy: 0.8289
Epoch 11/100
240/240 [==============================] - 4s 17ms/step - loss: 0.4870 - accuracy: 0.8374 - val_loss: 0.4899 - val_accuracy: 0.8332
Epoch 12/100
240/240 [==============================] - 4s 17ms/step - loss: 0.4796 - accuracy: 0.8401 - val_loss: 0.4806 - val_accuracy: 0.8352
Epoch 13/100
240/240 [==============================] - 4s 17ms/step - loss: 0.4719 - accuracy: 0.8421 - val_loss: 0.4744 - val_accuracy: 0.8368
Epoch 14/100
240/240 [==============================] - 4s 16ms/step - loss: 0.4662 - accuracy: 0.8431 - val_loss: 0.4726 - val_accuracy: 0.8346
Epoch 15/100
240/240 [==============================] - 4s 16ms/step - loss: 0.4597 - accuracy: 0.8460 - val_loss: 0.4654 - val_accuracy: 0.8383
Epoch 16/100
240/240 [==============================] - 4s 16ms/step - loss: 0.4550 - accuracy: 0.8468 - val_loss: 0.4598 - val_accuracy: 0.8414
Epoch 17/100
240/240 [==============================] - 4s 17ms/step - loss: 0.4501 - accuracy: 0.8482 - val_loss: 0.4563 - val_accuracy: 0.8424
```

В примере видно, что на начальных эпохах доля правильных ответов на проверочной выборке быстро увеличивается с 0.7548 на первой эпохе до 0.8294 на десятой эпохе. Но ближе к окончанию обучения доля правильных ответов на проверочной выборке почти не меняется. На 97-й эпохе она составляет 0.8764, затем снижается до 0.8740 (эпоха 98) и 0.8758 (эпоха 99), а после этого незначительно увеличивается до 0.8768 на сотой эпохе. Это означает, что мы близки к переобучению и обучение необходимо

останавливать.

Имея обученную нейросеть, следует разобраться с гиперпараметрами обучения.

Мы попытаемся улучшить качество обучения сети путем изменения гиперпараметров: количество эпох обучения, размер мини-выборки, количество нейронов во входном слое, количество скрытых слоев. Для этого проведем серию экспериментов, в каждом из которых будем менять один из гиперпараметров, и анализировать, как изменилось качество работы сети.

1. Количество эпох обучения. Оценим влияние количества эпох обучения на качество обучения сети. Количество эпох задается в аргументе epochs метода model.fit:

```
history = model.fit(x_train, y_train,
                    batch_size=200,
                    epochs=100,  # Количество эпох
                    validation_split=0.2,
                    verbose=1)
```

Попробуйте обучать сеть в течение 50, 75, 100 и 125 эпох. Выберите количество эпох, при котором самая высокая доля верных ответов нейросети на тестовых данных.

2. Размер мини-выборки. Оценим влияние размера мини-выборки на качество обучения сети. Размер задается в аргументе batch_size метода model.fit:

```
history = model.fit(x_train, y_train,
                    batch_size=200, # Размер мини-выборки
                    epochs=100,
                    validation_split=0.2,
                    verbose=1)
```

Используйте размер мини-выборки 50, 100, 200 и 400. Выберите значение, при котором самая высокая доля верных ответов нейросети на тестовых данных.

3. Количество нейронов входного слоя. Изменяйте количество нейронов во входном слое и оцените, как оно влияет на качество обучения сети. Количество нейронов задается при создании входного слоя:

```
model.add(Dense(XXX, input_dim=784, activation="relu"))
```

Используйте значения 500, 700, 900, 1200. Выберите значение, при котором самая высокая доля верных ответов нейросети на тестовых данных.

4. Добавляем скрытый слой. Добавим в нашу сеть скрытый слой, чтобы она стала глубокой:

```
model.add(Dense(800, input_dim=784, activation="relu"))
model.add(Dense(600, activation="relu"))# Новый скрытый слой
model.add(Dense(10, activation="softmax"))
```

Попробуйте добавить скрытый слой с разным количеством нейронов: 500, 700, 900 и 1200.

Выберите наиболее подходящее количество нейронов скрытого слоя. Оцените, как изменяется время обучения при добавлении скрытого слоя с разным количеством нейронов.

Контрольные вопросы:

1. Запишите долю верных ответов работы сети после обучения на тестовых данных.

2. Проанализируйте долю верных ответов на проверочном наборе данных в процессе обучения.

Лабораторная работа 3. Моделирование информационных процессов в среде ARENA

Arena – система имитационного моделирования, которая позволяет создавать динамические модели разнородных процессов и систем, оптимизировать построенную модель. Программа Arena снабжена удобным объектно-ориентированным интерфейсом, обладает широкими функциональными возможностями по адаптации к различным предметным областям.

Основой технологии моделирования Arena являются язык моделирования SIMAN и анимационная система Cinema Animation. тличается гибкими и выразительными средствами моделирования. Отображение результатов моделирования в Arena выполняется с использованием Cinema Animation.

Процесс моделирования организован следующим образом. Сначала пользователь, шаг за шагом, строит в визуальном редакторе программы Arena модель. Затем система генерирует по ней соответствующий код на SIMAN, после чего автоматически запускается Cinema Animation.

Arena состоит из блоков моделирования (модули) и операций (сущности). Сущности перемещаются между модулями по мере их обслуживания.

Построение простейшей имитационной модели:

1. Запустите программу Arena, выбрав Программы/Rockwell Software/Arena 9.0 из меню Пуск. Появится главное окно приложения (рис.1), которое содержит 3 области: окно рабочего модуля; окно свойств модулей;

окно проекта.

Окно проекта включает в себя несколько панелей:

– Basic Process (панель основных процессов);

– Reports (панель отчетов);

– Navigate (панель навигации).

В панели основных процессов Basic Process находятся основные графические модули и модули данных для создания простых имитационных моделей.

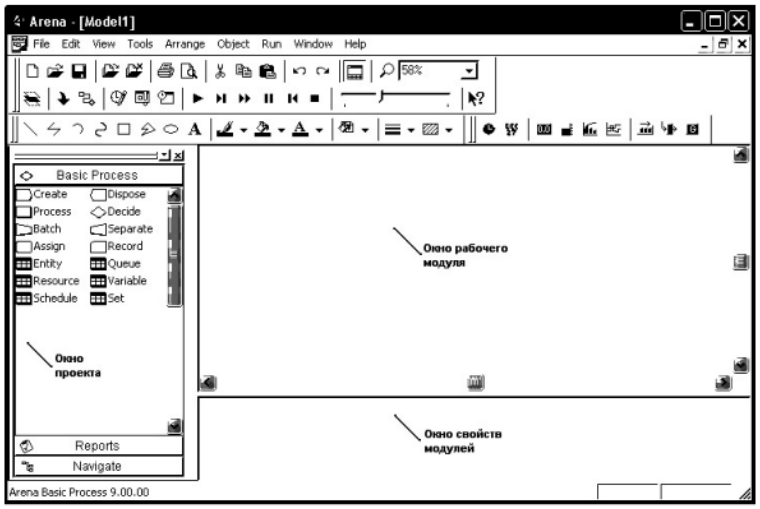
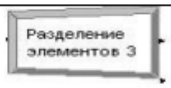
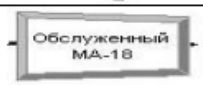


Рисунок 1- Главное окно программы Arena

Таблица 1.1. Основные модули панели Basic Process

	модуль *Create*. Является начальной точкой для элементов в имитационной модели. В модуле определяется тип элементов и время их появления.
	модуль *Separate*. Этот модуль используется для разделения элементов на отдельные составляющие.
	модуль *Process*. Основной модуль процесса обработки в имитационной модели. В модуле задаются ограничения на использование ресурсов, а также стоимость и временные характеристики процесса обработки.
	модуль *Decide*. Этот модуль позволяет читывать принятие решения в модели.
	модуль *Batch*. Этот модуль отвечает за механизм группировки в имитационной модели.
	модуль *Record*. Модуль предназначен для сбора статистики в имитационной модели.
	модуль *Dispose*. Этот модуль является выходной точкой из имитационной модели.

Модули помещаются в окно рабочего модуля методом «drug & drop», соединяются с помощью коннектора.

2. Построим простую имитационную модель на примере работы рабочей станции. Время поступления запросов в систему экспоненциально распределено со средним значением 30 минут, число запросов не ограничено, в случае занятости обсуживающегося устройства запрос встает в очередь. Время обслуживания запросов экспоненциально распределено со средним значением 24 минуты.

3. Переместите модули Create, Process и Dispose в окно рабочего модуля, как это показано на рисунке 2.

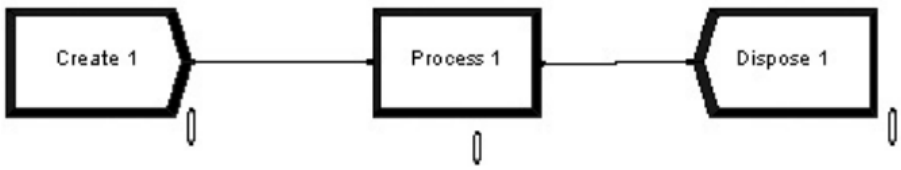


Рисунок 2- Имитационная модель работы рабочей станции

4. Для задания свойств графическому модулю необходимо дважды

щелкнуть по нему и в диалоге задать значения параметров (рис.3).

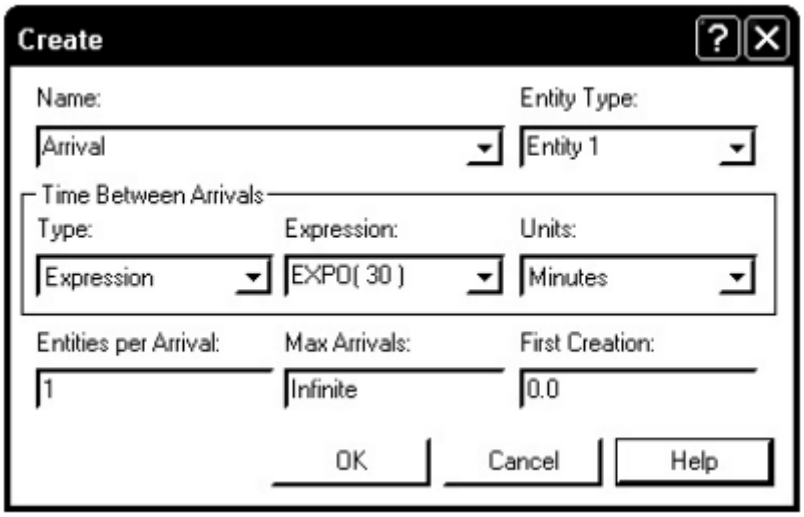


Рисунок 3- Диалоговое окно свойств модуля Create

Name – имя элемента.

Entity type – тип объекта (документы, товары, клиенты и т.д.).

Time between arrivals – время между объектами в потоке. Эта опция имеет несколько подопций, таких как:

Units – единицы измерения времени (секунды, минуты, часы, дни);

Type – тип временных промежутков, он может быть: Constant (постоянный); Random (случайный по экспоненциальному закону); Expression (выраженный каким-либо видом распределения: нормальное, пуассоновское, логнормальное и т.д.)

Max Arrivals (Максимальный размер потока).

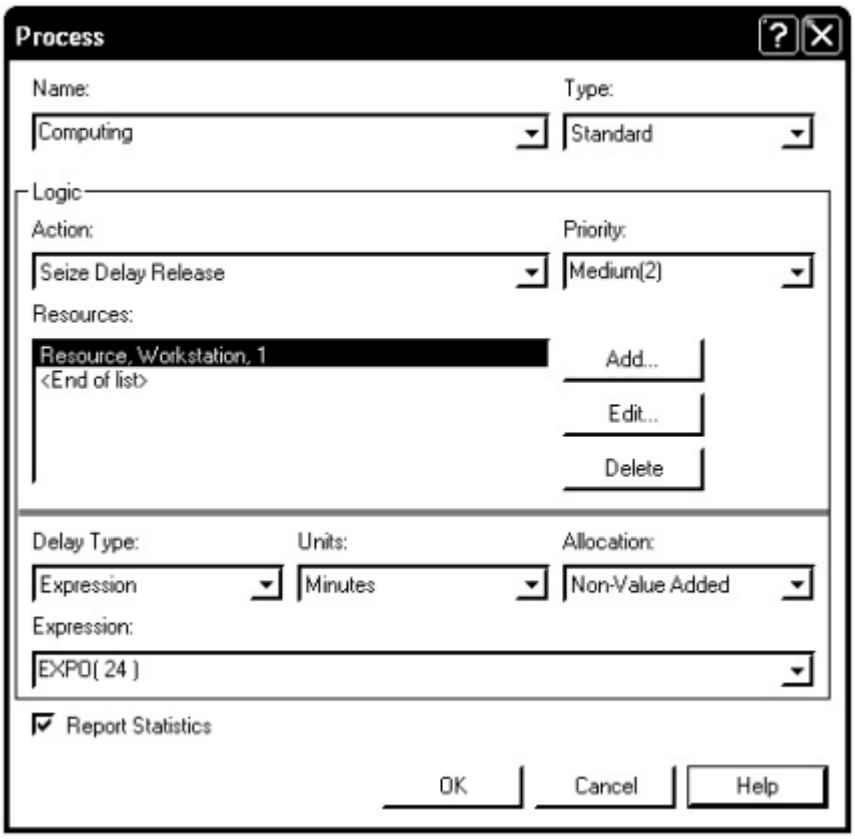


Рисунок 4- Диалоговое окно свойств модуля Process

Поле Resources определяет ресурсы или группы ресурсов, которые будут обрабатывать сущности в этом модуле. Добавление ресурса кнопкой Add, в появившемся окне (рис.5) указать использование одного ресурса.

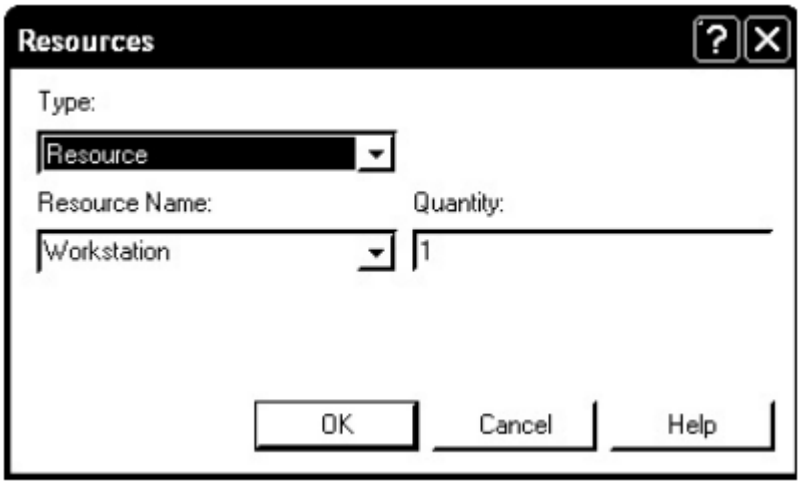


Рисунок 5- Диалоговое окно задания ресурсов в модуле Process

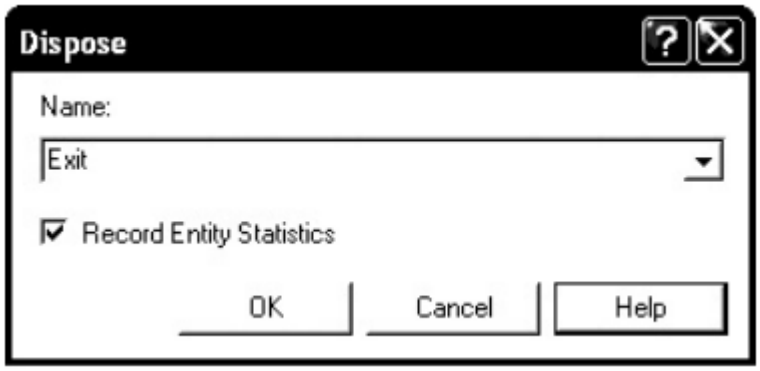


Рисунок 6- Диалоговое окно свойств модуля Dispose

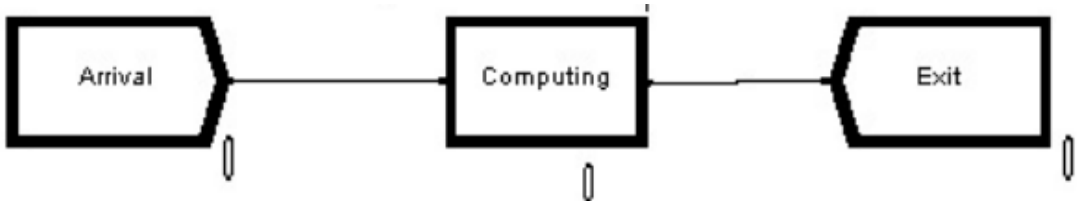


Рисунок 7 - Имитационная модель работы рабочей станции

Пример. В качестве примера приведена имитационная модель ТО маслоагрегата МА-18 двигателя Д-18 (рис. 8 –10), построенная на основе IDEF3 и DFD моделей.

Рисунок 8 - Верхний уровень имитационной модели технического обслуживания маслонасоса МА-18

Рисунок 9 - Подуровень имитационной модели технического обслуживания

маслонасоса МА-18

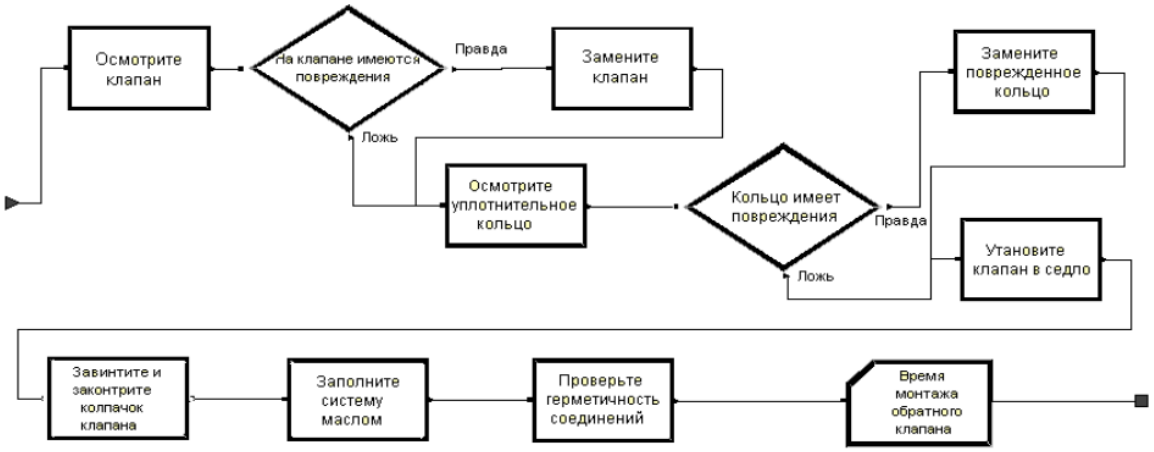


Рисунок 10 - Имитационная модель процесса технического обслуживания обратного клапана маслонасоса

По результатам имитационного моделирования проведен анализ производственного процесса ТО маслонасоса МА-18. Получены временные характеристики производственного процесса в зависимости от наличия свободных ресурсов и возникающих ситуаций входе ТО.

На рисунке 11 представлен результат расчета времени затрат на ТО одного изделия. Для получения устойчивой статистики проведены расчеты временных затрат на ТО 1000 изделий, результаты расчета показаны на рисунке 12.

Время выполнения операций

Общее время на каждый объект	Средняя величина	Минимальное значение	Максимальное значение
Демонтаж и монтаж МА-18	1.3595	1.2912	1.4463
Демонтаж и монтаж обратного клапана МА-18	0.6955	0.6546	0.7423
Демонтаж МА-18	0.6336	0.6002	0.6626
Демонтаж обратного клапана	0.3088	0.2843	0.3325
Монтаж МА-18	0.7260	0.6627	0.7965
Монтаж обратного клапана	0.3867	0.3547	0.4226
Регулировка перепада давления на входе в двигатель	0.2142	0.2036	0.2240
ТО маслоагрегата МА-18	2.2855	2.1494	2.4216

Рисунок 11 - Временные затраты на каждую операцию при обслуживании одного изделия

Из рисунков 11 и 12 можно определить на какую операцию по ТО маслоагрегата затрачивается наибольшее количество времени и, проанализировав ее, можно предложить мероприятия по сокращению её продолжительности, с учетом сохранений требований по обеспечению безопасности полетов.

ТО маслоагрегата

Повторение 1 Единица времени: Часы

Процесс

Сумарное время

	Величина
Демонтаж и монтаж МА-18	1358.14
Демонтаж и монтаж обратного клапана МА-18	695.46
Демонтаж МА-18	633.55
Демонтаж обратного клапана	308.79
Монтаж МА-18	725.23
Монтаж обратного клапана	386.67
Регулировка перепада давления на входе в двигатель	214.23
ТО маслоагрегата МА-18	2265.74

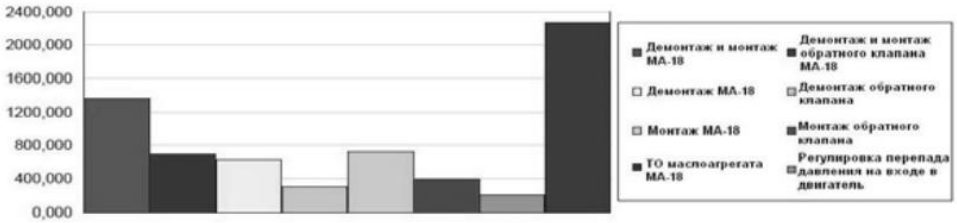


Рисунок 12 - Расчет временных характеристик по имитационной модели

Задание. Смоделировать работу системы технического обслуживания воздушного судна по указанию преподавателя.

Контрольные вопросы:

1. В чем особенность имитационного моделирования?

2. Какова основная цель имитационного моделирования?

3. Назовите основные части простой имитационной модели СМО в пакете Arena?

4. Какие возможности предоставляет пакет Arena для проектирования имитационных моделей?

Лабораторная работа 4. Определение потребностей в CASE-средствах

Цель работы: определить потребности в CASE-средствах для компании в рамках выбранной предметной области.

Этап определения потребности в CASE-средствах (рис.1) включает достижение понимания потребностей организации и технологии последующего процесса внедрения CASE-средств. Он должен привести к выделению тех областей деятельности организации, в которых применение CASE-средств может принести реальную пользу. Результатом данного этапа является документ, определяющий стратегию внедрения CASE-средств.

Рисунок 1- Определение потребностей в CASE-средствах

Задание:

Разработать документ, определяющий стратегию внедрения CASE-средств для компании в рамках выбранной предметной области.

Список использованных источников

1. C.C.Aggarwal. Neural Networks and Deep Learning. A Textbook. Springer International Publishing AG, 2018. DOI 10.1007/978-3-319-94463-0 ISBN 978-3-319-94462-3
2. Акопов, А.С. Имитационное моделирование. учебник и практикум для академического бакалавриата / А.С. Акопов. - Люберцы: Юрайт, 2016. - 389 с.
3. Варжапетян, А.Г. Имитационное моделирование на GPSS/H / А.Г. Варжапетян. - М.: Вузовская книга, 2007. - 424 с.
4. военной истории / Н.В. Митюков. - М.: Изд. ЛКИ, 2011. - 280 с.
5. Вьюненко, Л.Ф. Имитационное моделирование. учебник и практикум для академического бакалавриата / Л.Ф. Вьюненко, М.В. Михайлов, Т.Н. Первозванская. - Люберцы: Юрайт, 2016. - 283 с.
6. Галушкин, А.И. Нейронные сети: история развития теории: Учебное пособие для вузов. / А.И. Галушкин, Я.З. Цыпкин. - М.: Альянс, 2015. - 840 с.
7. Галушкин, А.И. Нейронные сети: основы теории. / А.И. Галушкин. - М.: ГЛТ, 2010. - 496 с.
8. Девятков, В.В. Имитационное моделирование: Учебное пособие / Н.Б. Кобелев, В.А. Половников, В.В. Девятков. - М.: КУРС, НИЦ ИНФРА-М, 2013. - 368 с.
9. Каллан, Р. Нейронные сети: Краткий справочник / Р. Каллан. - М.: Вильямс И.Д., 2017. - 288 с.
10. Карпов, Ю. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5 / Ю. Карпов. - СПб.: BHV, 2009. - 400 с.

11. Лычкина, Н.Н. Имитационное моделирование экономических процессов: Учебное пособие / Н.Н. Лычкина. - М.: НИЦ ИНФРА-М, 2012. - 254 с.

12. Решмин, Б.И. Имитационное моделирование и системы управления / Б.И. Решмин. - Вологда: Инфра-Инженерия, 2016. - 74 с.

13. С.А.Шумский. Машинный интеллект. Очерки по теории машинного обучения и искусственного интеллекта. М., РИОР, 2019. DOI: 10.29039/02011-1