Министерство по образованию и науке РФ
Московский государственный горный университет

КАФЕДРА ЭЛЕКТРОТЕХНИКИ И ИНФОРМАЦОННЫХ СИСТЕМ

Утверждён Советом
по методической работе и
качеству образования

Танцов П.Н.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

по дисциплине

ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

для направлений подготовки
230200 – Информационные системы
230201 – Информационные системы и технологии

Москва, 2013

УДК

Содержание

Введение

В настоящее время у людей есть потребность в создании систем, способных решать задачи, не имеющие чёткого алгоритма решения, такие, например, как задачи распознавания, классификации, анализа, логико-лингвистические задачи. В качестве инструмента для решения подобных задач целесообразно применять искусственные нейронные сети, а также алгоритмы нечёткого ввода-вывода.

Настоящее пособие предназначено для приобретения студентами некоторых навыков работы с искусственными нейронными сетями с целью дальнейшего применения их в научно-практической деятельности.

Пособие состоит из 3-х частей. Первая часть содержит указания по применению на практике классических искусственных нейронных сетей, построенных на простейшей математической модели нейрона мозга человека Макалока-Питтса. В качестве учебных примеров даны задачи «адекватного реагирования» и распознавания зрительных образов.

Вторая часть состоит из лабораторных работ по применению нейросетевой парадигмы, разработанной советским и российским учёным В.Д. Цыганковым. Как и классические нейронные сети, основанные на парадигме Маккалока-Питтса, подход В.Д. Цыганкова также успешно применяется в науке и технике.

Третья часть содержит одну лабораторную работу с рекомендациями по созданию нечётких экспертных систем, применение которых целесообразно, например, для задач анализа, прогнозирования, управления различными объектами или процессами.

Лабораторные работы 1-6 целесообразно выполнять на языке программирования высокого уровня, что даст возможность студентам лучше понять технологию применения нейросетевых алгоритмов. Выполнение лабораторной работы №7 возможно как на языке программирования, так и в среде «МатЛаб».

Автор выражает свою благодарность В.Д. Цыганкову и И.В. Степаняну за помощь в предоставлении материалов для лабораторных работ.

Лабораторная работа №1
НАСТРОЙКА ПАРАМЕТРОВ НЕЙРОНА С ПОРОГОВОЙ ФУНКЦИЕЙ АКТИВАЦИИ

Цель работы: знакомство с правилами настройки параметров нейрона по алгоритму Розенблатта.

Теоретическая часть

Человеческий мозг содержит свыше тысячи миллиардов вычислительных элементов, называемых *нейронами.* Превышая по количеству число звезд в Млечном Пути галактики, эти нейроны связаны сотнями триллионов нервных нитей, называемых *аксонами*. Эта сеть нейронов отвечает за все явления, которые мы называем мыслями, эмоциями, познанием, а также и за совершение мириадов сенсомоторных и автономных функций. Пока мало понятно, каким образом все это происходит, но уже исследовано много вопросов физиологической структуры и определенные функциональные области постепенно изучаются исследователями.

Биологический нейрон

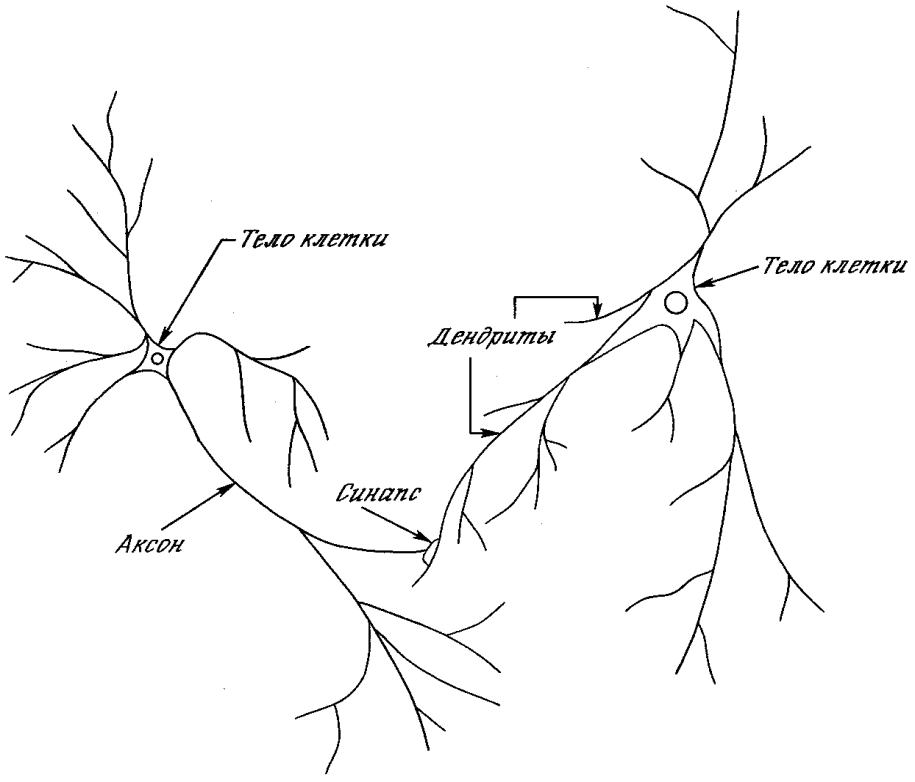


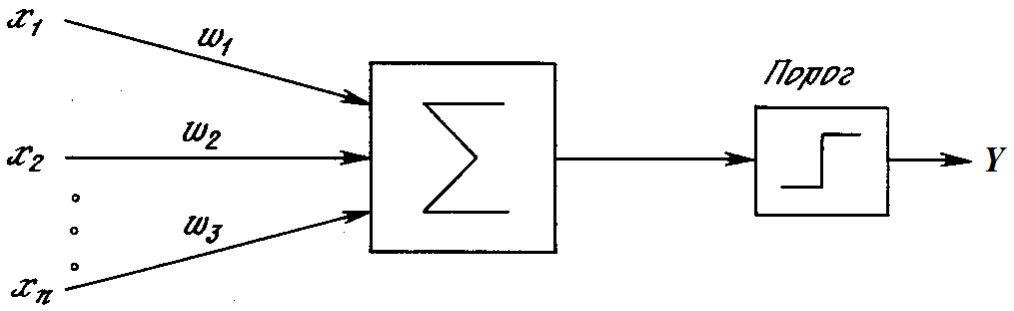
Рис. 1.1. Строение нейрона

Нейрон является основным строительным блоком нервной системы. Он. является клеткой, подобной всем другим клеткам тела; однако определенные существенные отличия позволяют ему выполнять все вычислительные функции и функции связи внутри мозга.

Как показано на рис. 1.1, нейрон состоит из трех частей: тела клетки, дендритов и аксона, каждая часть со своими, но взаимосвязанными функциями. Дендриты идут от тела нервной клетки к другим нейронам, где они принимают сигналы в точках соединения, называемых синапсами. Принятые синапсом входные сигналы подводятся к телу нейрона. Здесь они суммируются, причем одни входы стремятся возбудить нейрон, другие – воспрепятствовать его возбуждению. Когда суммарное возбуждение в теле нейрона превышает некоторый порог, нейрон возбуждается, посылая по аксону сигнал другим нейронам. У этой основной функциональной схемы много усложнений и исключений, тем не менее, большинство искусственных нейронных сетей моделируют лишь эти простые свойства.

Искусственный нейрон

Стремясь воспроизвести функции человеческого мозга, исследователи создали простые аппаратные (а позже программные) модели биологического нейрона и системы его соединений. Когда нейрофизиологи достигли более глубокого понимания нервной системы человека, эти ранние попытки стали восприниматься как весьма грубые аппроксимации. Тем не менее на этом пути были достигнуты впечатляющие результаты, стимулировавшие дальнейшие исследования, приведшие к созданию более изощренных сетей.



Конец

Первое систематическое изучение искусственных нейронных сетей было предпринято Маккалокком и Питтсом в 1943 г. [1]. Позднее в работе [2] они

исследовали сетевые парадигмы для распознавания изображений, подвергаемых сдвигам и поворотам. Простая нейронная модель, показанная на рис. 1.2, использовалась в большей части их работы. Элемент **Σ** умножает каждый вход x на вес w и суммирует взвешенные входы. Если эта сумма больше заданного порогового значения, выход равен единице, в противном случае – нулю. Эти системы (и множество им подобных) получили название *персептронов.* Они состоят из одного слоя искусственных нейронов, соединенных с помощью весовых коэффициентов с множеством входов, хотя в принципе описываются и более сложные системы**.**

В 60-е годы персептроны вызвали большой интерес и оптимизм. Розенблатт [3] доказал замечательную теорему об обучении персептронов. Доказательство этой теоремы показало, что персептрон способен научиться всему, что он способен представлять. Важно при этом уметь различать представляемость и обучаемость. Понятие представляемости относится к способности персептрона (или другой сети) моделировать определенную функцию. Обучаемость же требует наличия систематической процедуры настройки весов сети для реализации этой функции.

Математическая модель

На вход искусственного нейрона поступает некоторое множество сигналов, каждый из которых является выходом другого нейрона. Каждый вход умножается на соответствующий вес, аналогичный синаптической силе, и все произведения суммируются, определяя уровень активации нейрона. На рис. 1.3 представлена модель, реализующая эту идею. Хотя сетевые парадигмы весьма разнообразны, в основе почти всех их лежит эта конфигурация. Здесь множество входных сигналов, обозначенных x_1, x_2,…, x_n, поступает на искусственный нейрон. Эти входные сигналы, в совокупности обозначаемые вектором **X**, соответствуют сигналам, приходящим в синапсы биологического нейрона. Каждый сигнал умножается на соответствующий вес w_1, w_2,…, w_n, и поступает на суммирующий блок, обозначенный **Σ**. Каждый вес соответствует «силе» одной биологической синаптической связи. Множество весов в

совокупности обозначается вектором **W**. Суммирующий блок, соответствующий телу биологического элемента, складывает взвешенные входы алгебраически, создавая выход, который мы будем называть **S**. В векторных обозначениях это может быть компактно записано следующим образом:

$$S = XW$$

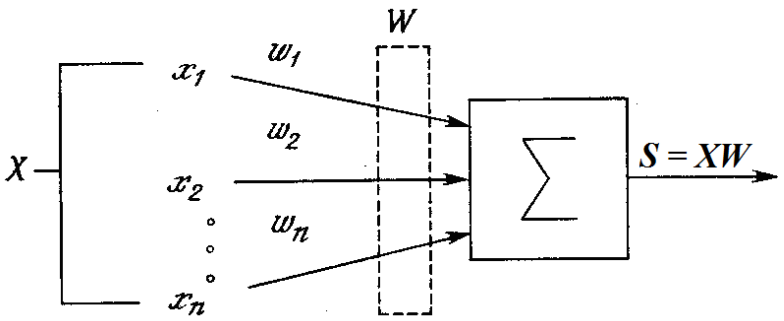


Рис. 1.3. Искусственный нейрон

Активационная функция

Сигнал **S** далее, как правило, преобразуется активационной функцией *F(S)* и дает выходной нейронный сигнал **Y** (рис. 1.4). Активационная функция может быть обычной линейной функцией

$$Y = kS,$$

где k – постоянная, пороговой функцией

$$Y = 1, \text{ если } S > \tau,$$

$$Y = 0 \text{ в остальных случаях,}$$

где τ – некоторая постоянная пороговая величина, или же функцией, более точно моделирующей нелинейную передаточную характеристику биологического нейрона и представляющей нейронной сети большие возможности.

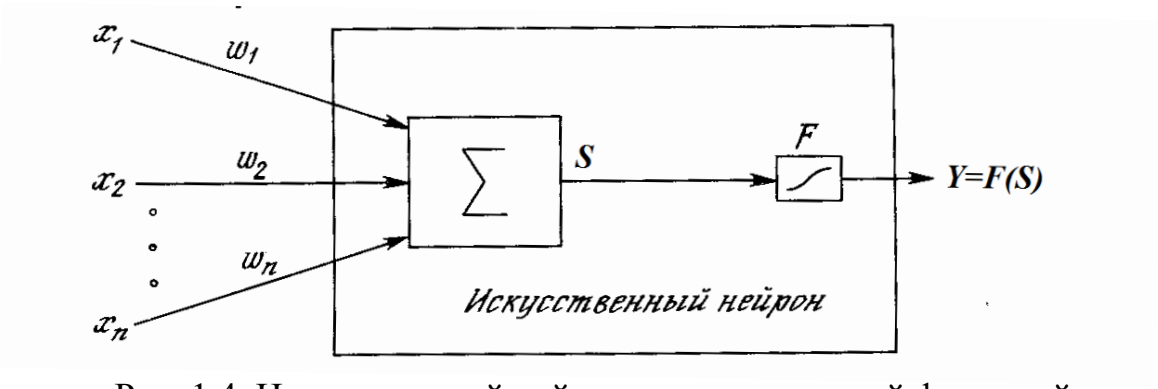


Рис. 1.4. Искусственный нейрон с активационной функцией

На рис. 1.4 блок, обозначенный как *F*, принимает сигнал **S** и выдает сигнал **Y**. Если блок *F* сужает диапазон изменения величины **S** так, что при любых значениях **S** значения **Y** принадлежат некоторому конечному интервалу, то *F* называется *«сжимающей» функцией.* В качестве «сжимающей» функции часто используется логистическая или «сигмоидальная» (S-образная) функция, показанная на рис. 1.5. Эта функция математически выражается как

$$Y = \frac{1}{1 + e^{-\alpha S}}$$

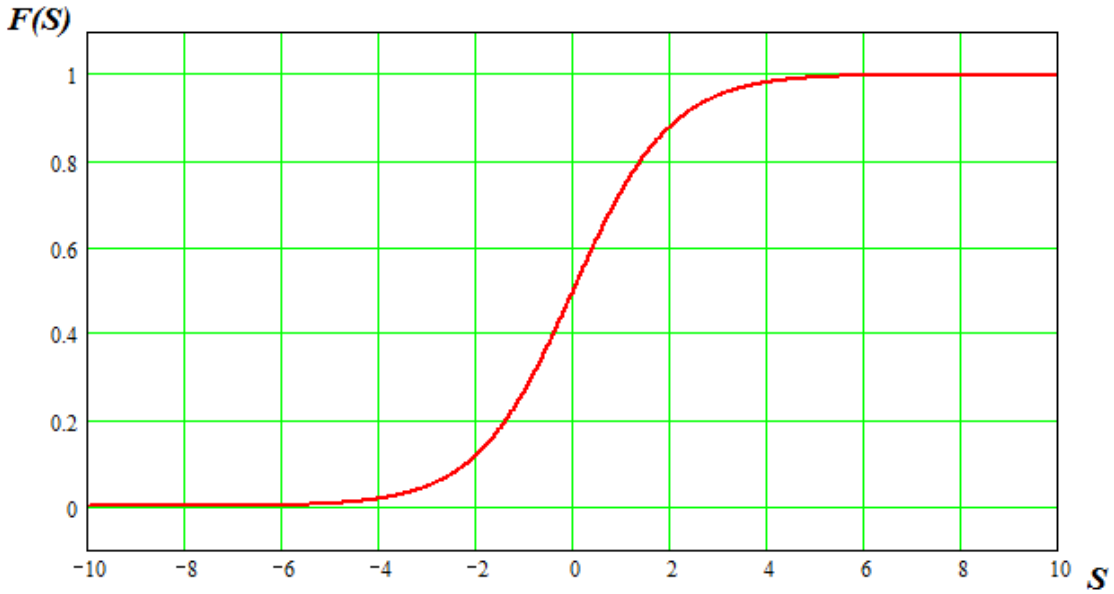


Рис. 1.5. Сигмоидальная логистическая функция

Параметр $\alpha \geq 0$ влияет на форму кривой. Если α очень велико, то функция превращается в пороговую, если α близко к нулю, то функция превращается в горизонтальную прямую *Y*=0,5.

Другой широко используемой активационной функцией является гиперболический тангенс. По форме она сходна с логистической функцией и часто используется биологами в качестве математической модели активации нервной клетки. В качестве активационной функции искусственной нейронной сети она записывается следующим образом:

$$Y = th(S)$$

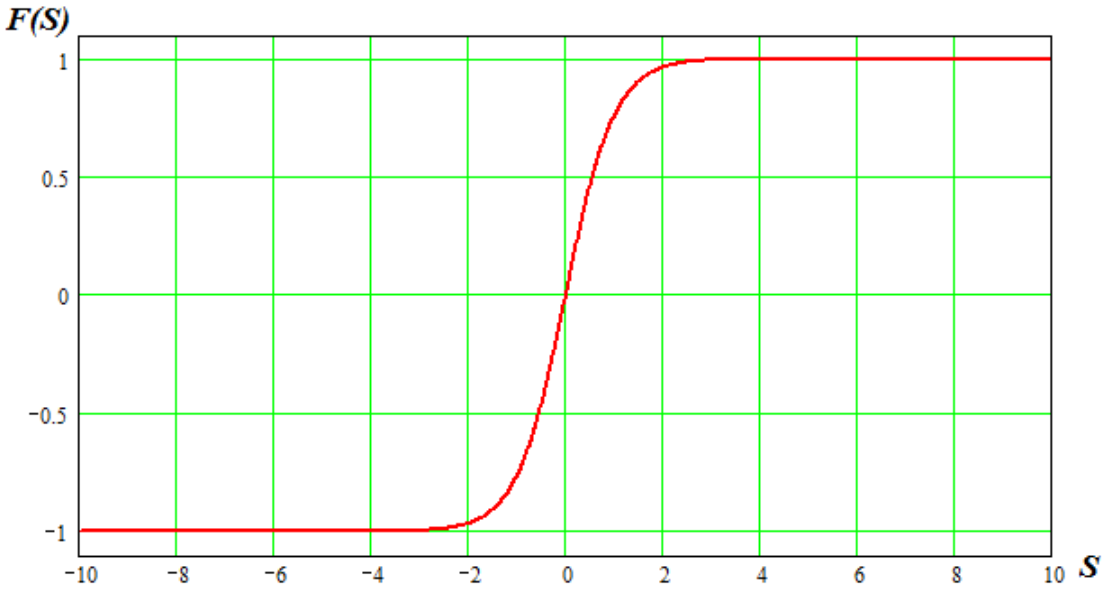


Рис. 1.6. Функция гиперболического тангенса

Подобно логистической функции гиперболический тангенс является S-образной функцией, но он симметричен относительно начала координат, и в точке **S** = 0 значение выходного сигнала **Y** равно нулю (см. рис. 1.6). В отличие от логистической функции гиперболический тангенс принимает значения различных знаков, что оказывается выгодным для ряда сетей.

Практическая часть

Задание: Создать модель персептрона с одним нейроном, ступенчатой функцией активации; обучить нейрон на обучающем множестве, состоящей из 4-х пар вход-цель, при помощи алгоритма Розенблатта.

$$\left\{ x_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix},\ t_1 = 0 \right\} \qquad \left\{ x_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix},\ t_2 = 1 \right\}$$

$$\left\{x_3 = \begin{bmatrix} -2 \\ 2 \end{bmatrix},\ t_3 = 0\right\} \qquad \left\{x_4 = \begin{bmatrix} -1 \\ 2 \end{bmatrix},\ t_4 = 1\right\}$$

Порядок выполнения работы:

Данная задача представляет собой задачу классификации. В результате обучения нейрон должен правильно реагировать на заданные входные вектора. Входные векторы содержат по 2 элемента, следовательно, нейрон будет иметь 2 входа (рис. 1.7). Пороговая функция определяется задаётся следующим соотношением:

$$\begin{cases} F(S) = 0, & \textit{если} \quad S < 0 \\ F(S) = 1, & \textit{если} \quad S \geq 0 \end{cases}$$

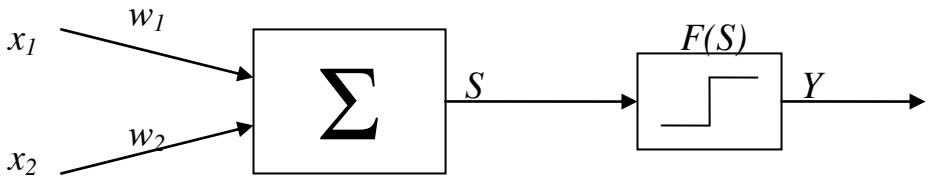


Рис. 1.7. Нейрон с двумя входами и пороговой функцией активации

Рассмотрим алгоритм обучения персептрона Розенблатта:

Начальные значения весов принимаем за 0. Вычислим выход нейрона для первого вектора x_1:

$$S = \sum_{i=1}^{2} x_{1i} \cdot w_i = x_{11} \cdot w_1 + x_{12} \cdot w_2 = 0$$

$$Y = F(0) = 1$$

Выход нейрона не совпадает с целевым значением t_1, поэтому необходимо применить правило настройки весов нейрона:

$$e = t_1 - Y = -1$$

$$\Delta w = e \cdot x_1 = -1 \cdot \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

Новые значения весов определяются по формуле

$$w^{new} = w^{old} + \Delta w = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -2 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \end{bmatrix}.$$

Далее на вход нейрона подаются поочерёдно вектора x_2 , x_3, x_4, для каждого из них вычисляется выход нейрона **Y**, и если выход не совпадает с соответствующим целевым значением, осуществляется процедура настройки весов.

Обучение нейрона можно считать оконченным тогда, когда для каждого вектора из обучающего множества выход нейрона совпадает с соответствующим целевым значением.

Отчёт должен содержать:

1. Задание.
2. Ход выполнения работы.
3. Программный код или блок-схема алгоритма обучения.
4. Результат работы: конечные значения весов и количество циклов обучения, потребовавшееся для решения задачи.
5. Практическое задание: придумайте такое обучающее множество, то есть значения пар вход-цель, и начальные значения весов, чтобы нейрон невозможно было обучить при помощи алгоритма Розенблатта.

Контрольные вопросы:

1. Строение биологического нейрона.
2. Математическая модель нейрона.
3. Активационная функция нейрона.
4. Алгоритм обучения Розенблатта.

Литература:

1. McCulloch W. W., Pitts W. 1943. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5:115-33.

2. Pitts W. Moculloch W. W. 1947. How we know universals. Bulletin of Mathematical Biophysics 9:127-47.

3. Rosenblatt F. 1962. Principles of Neurodynamics. New York: Spartan Books.

4. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика, М.: «Мир», 1992, 184 с.

Лабораторная работа №2

РАСПОЗНАВАНИЕ ОБРАЗОВ ПОСРЕДСТВОМ СЕТЕЙ С МЕТОДОМ ОБУЧЕНИЯ ОБРАТНОГО РАСПРОСТРАНЕНИЯ

Цель работы: знакомство с процедурой обратного распространения ошибки, моделирование нейронной сети для распознавания зрительных образов.

Теоретическая часть

Долгое время не было теоретически обоснованного алгоритма для обучения многослойных искусственных нейронных сетей. Разработка алгоритма обратного распространения сыграла важную роль в возрождении интереса к искусственным нейронным сетям. Обратное распространение – это систематический метод для обучения многослойных искусственных нейронных сетей. Несмотря на некоторые ограничения, процедура обратного распространения сильно расширила область проблем, в которых могут быть использованы искусственные нейронные сети, и убедительно продемонстрировала свою мощь.

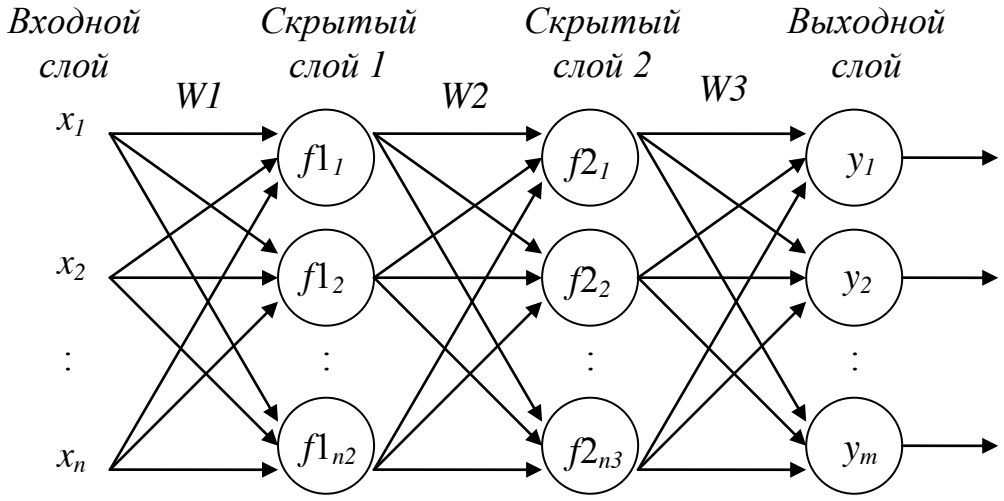


Рис. 2.1. Многослойная сеть обратного распространения

Процедура обратного распространения применима к сетям с любым числом слоев. Продемонстрируем процедуру обратного распространения на трёхслойной сети (рис. 2.1). Количество нейронов в каждом слое может быть

любым, и для каждой конкретной задачи выбирается оптимальный вариант. Для обучения сети задаётся обучающее множество (совокупность пар векторов вход-цель).

Перед началом обучения всем весам должны быть присвоены небольшие начальные значения, выбранные случайным образом. Это гарантирует, что в сети не произойдет насыщения большими значениями весов, и предотвращает ряд других патологических случаев. Например, если всем весам придать одинаковые начальные значения, а для требуемого функционирования нужны неравные значения, то сеть не сможет обучиться.

Обучение сети обратного распространения требует выполнения следующих операций:

1. Выбираем очередную обучающую пару из обучающего множества; входной вектор подаём на вход сети.

2. Вычисляем выход сети:

$$f1_i = F\left(\sum_{j=1}^{n} w1_{ji} x_j\right), \qquad i = 1,\ldots,n2$$

$$f2_i = F\left(\sum_{j=1}^{n2} w2_{ji} f1_j\right), \qquad i = 1,\ldots,n3$$

$$y_i = F\left(\sum_{j=1}^{n3} w3_{ji} f2_j\right), \qquad i = 1,\ldots,m$$

$$F(S) = \frac{1}{1+e^{-S}}$$

3. Для каждого нейрона выходного слоя вычисляем ошибку, которая равна разности между целевым и выходным значениями:

$$\delta(y_i) = t_i - y_i, \qquad i = 1,2,\ldots,m$$

Здесь же вычисляем среднеквадратичную погрешность e, которая равна

$$e = \frac{1}{2}\sum_{i=1}^{m}\delta(y_i)^2$$

4. Производим корректировку весов от скрытого слоя 2 к выходному слою по формуле:

$$w3_{ij}^{new} = w3_{ij}^{old} + \eta\delta(y_i)\frac{dy_i}{dS_i}f2_i$$

$$i = 1,2,\ldots,n3, \quad j = 1,2,\ldots,m$$

Коэффициент η задаётся в диапазоне от 0 до 1 и влияет на скорость обучения сети.

$\frac{dy_i}{dS_i}$ есть производная от функции активации. Если активационной функцией является сигмоидальная функция $F(S) = \frac{1}{1+e^{-S}}$, то её производная $\frac{dF(S)}{dS} = F(S)(1-F(S))$, и формула корректировки весов будет выглядеть следующим образом:

$$w3_{ij}^{new} = w3_{ij}^{old} + \eta\delta(y_j)y_j(1-y_j)f2_i$$

$$i = 1,2,\ldots,n3, \quad j = 1,2,\ldots,m$$

5. Переходим на один уровень влево. Вычисляем ошибки для нейронов 2-го скрытого слоя. Они определяются как взвешенные суммы ошибок выходного слоя:

$$\delta(f2_i) = \sum_{j=1}^{m}\delta(y_j)w3_{ij}, \; i = 1,2,\ldots,n3$$

6. Корректируем веса связей между нейронами 1-го и 2-го слоя:

$$w2_{ij}^{new} = w2_{ij}^{old} + \eta\delta(f2_j)f2_j(1-f2_j)f1_i$$

$$i = 1,2,\ldots,n2, \quad j = 1,2,\ldots,n3$$

7. Вычисляем ошибки для нейронов 1-го уровня:

$$\delta(f1_i) = \sum_{j=1}^{n3} \delta(f2_j) w2_{ij}\ , \qquad i = 1,2,\ldots,n2$$

8. Корректируем веса связей между входами и нейронами 1-го слоя:

$$w1_{ij}^{new} = w1_{ij}^{old} + \eta\delta(f1_j) f1_j (1 - f1_j) x_i$$

$$i = 1,2,\ldots,n, \quad j = 1,2,\ldots,n2$$

9. Проделываем п. 1-8 для каждого элемента из обучающего множества.

10. Проделываем п. 1-9 до тех пор, пока максимальное значение погрешности e не станет меньше изначально заданного порога ε.

Трудности, связанные с алгоритмом обратного распространения

Стоит отметить, что общепринятый от 0 до 1 динамический диапазон входов и выходов скрытых нейронов неоптимален. Так как величина коррекции веса пропорциональна выходному уровню нейрона, то нулевой уровень ведет к тому, что вес не меняется. При двоичных входных векторах половина входов в среднем будет равна нулю, и веса, с которыми они связаны, не будут обучаться! Решение состоит в приведении входов к значениям ±½ и добавлении смещения к сжимающей функции, чтобы она также принимала значения ±½. Новая сжимающая функция выглядит следующим образом:

$$\mathrm{F}(S) = -1/2 + \frac{1}{1 + \mathrm{e}^{-S}}$$

С помощью таких простых средств время сходимости сокращается в среднем от 30 до 50%. Это является одним из примеров практической модификации, существенно улучшающей характеристику алгоритма.

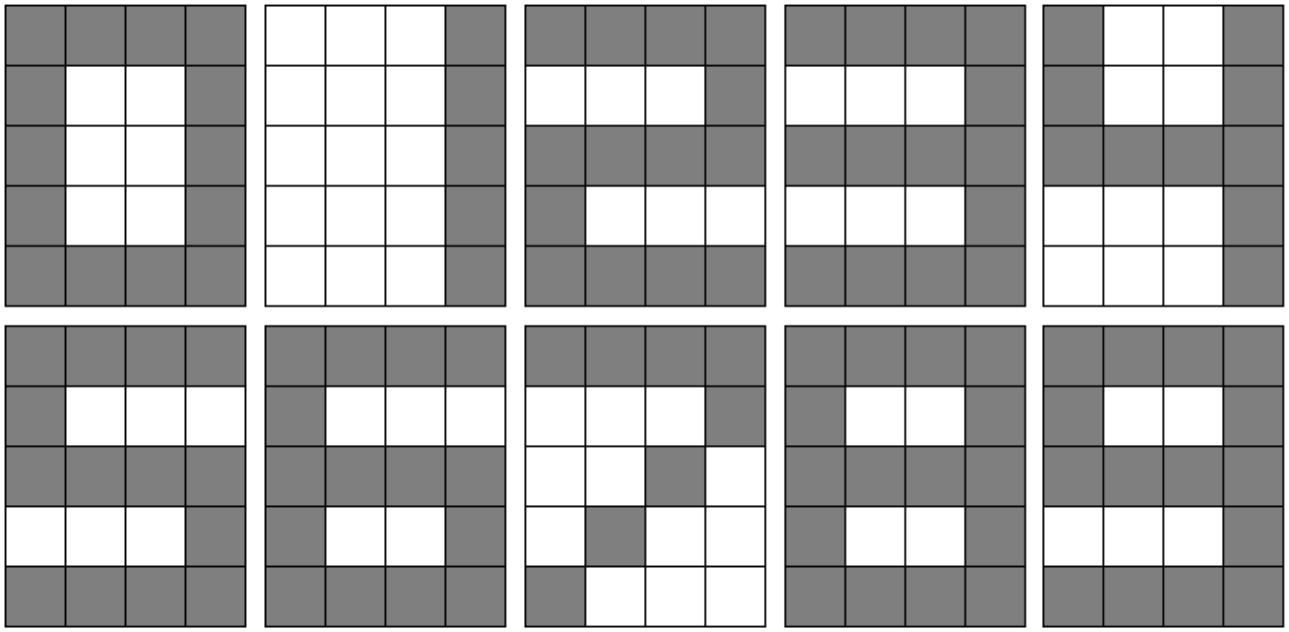
Также обратим внимание, что в процессе обучения сети значения весов могут в результате коррекции стать очень большими величинами. Это может привести к тому, что все или большинство нейронов будут функционировать при очень больших выходных значениях, в области, где производная сжимающей функции очень мала. Так как посылаемая обратно в процессе обучения ошибка пропорциональна этой производной, то процесс обучения может практически замереть. В теоретическом отношении эта проблема плохо

изучена. Обычно этого избегают уменьшением размера шага η, но это увеличивает время обучения.

Практическая часть

Задание: Создать многослойную нейронную сеть; обучить её распознавать зрительные образы арабских цифр посредством алгоритма обратного распространения ошибки.

Определим образы арабских цифр. Каждая цифра проецируется на двухцветную сенсорную матрицу размером 4x5 пикселей. Запишем образы в текстовый файл в бинарном виде: 0 – светлый пиксель, 1 – тёмный пиксель. После каждого образа указывается его целевое значение (рис 2.2).



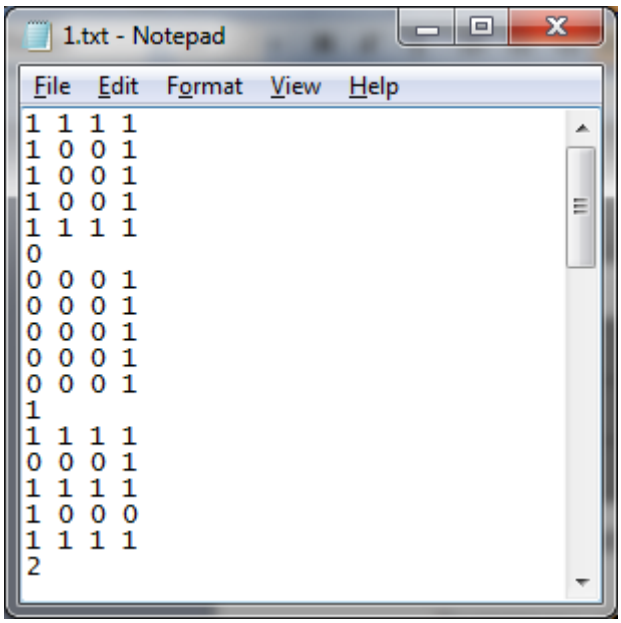


Рис. 2.2. Зрительные образы арабских цифр

Для данной конкретной задачи создаваемая нейронная сеть должна иметь 20 входов (по количеству элементов сенсорной матрицы) и 10 нейронов в выходном слое (по количеству элементов в обучающем множестве). Количество скрытых слоёв и количество нейронов в каждом из них подберите самостоятельно.

Целевое значение, стоящее в текстовом файле сразу после образа показывает, какой нейрон должен активироваться для этого образа. Для цифры «0» должен активироваться 0-й нейрон (для удобства нумерацию будем вести от 0 до 9); вектор целевых значений **T** в этом случае будет выглядеть так:

$$T = \{1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\},$$

Так как на практике обучить нейронную сеть до такой степени, чтобы она выдавала такие значения на выходах, невозможно, при проверке работы сети выбирается наибольшее значение на выходе среди всех нейронов выходного слоя. То есть если, например, 2-й нейрон имеет наибольшее значение из всех, значит, сеть склоняется к тому, что на вход поступил образ цифры «2». Это и будет ответом сети.

Максимально допустимая среднеквадратичная погрешность в результате обучения должна быть меньше 0.1 для каждого элемента обучающего множества.

Отчёт должен содержать:

1. Задание.
2. Ход выполнения работы.
3. Программный код или блок-схема алгоритма обучения.
4. График зависимости величины среднеквадратичной погрешности от числа итераций.
5. Пример работы программы: подайте на вход сети любой образ из обучающего множества и выведите значения нейронов выходного слоя.

Контрольные вопросы:

1. Схема многослойной нейронной сети для распознавания образов.
2. Описание алгоритма обратного распространения ошибки.
3. Формула корректировки весов в алгоритме обратного распространения.
4. Способы повышения сходимости процедуры обучения при использовании алгоритма обратного распространения.

Литература:

1. Werbos P. J. 1974. Beyond regression: New tools for prediction and analysis in the behavioral sciences. Masters thesis, Harward University.
2. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика, М.: «Мир», 1992, 184 с.
3. http://robocraft.ru/blog/algorithm/560.html
4. http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Лабораторная работа №3

ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ В УСЛОВИЯХ ШУМА (ПРОДОЛЖЕНИЕ ЛАБОРАТОРНОЙ РАБОТЫ №2)

Цель работы: освоение метода обучения нейронной сети в условиях зашумлённости входных данных.

Практическая часть

Задание: Обучить нейронную сеть распознавать зашумлённые образы.

В предыдущей лабораторной работе мы задавали образы цифр в бинарном виде – 0 – светлый пиксель, 1 – тёмный пиксель. Теперь к значению каждому пикселя будем добавлять случайную величину, которая задаётся равномерным распределением в диапазоне от –n до n (рис. 3.1.). Значение *n* будем называть уровнем шума.

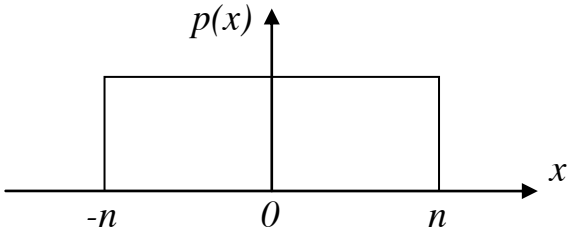
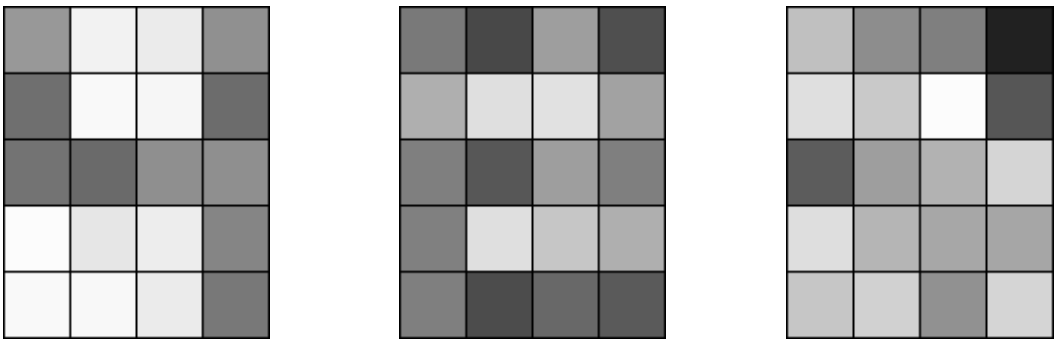


Рис. 3.1. Гистограмма распределения шума

На рис. 3.2 представлены образы цифр 4, 8, 9 соответственно с уровнем шума 0.2, 0.5 и 0.8. Чем больше уровень шума, тем хуже распознаётся образ.



а *б* *в*

Рис. 3.2. Примерный вид зашумлённых символов: *а* – уровень шума 0.2; *б* – уровень шума 0.5; *в* – уровень шума 0.8

Порядок выполнения работы:

Обучение нейронной сети в условиях шума будет мало отличаться от обучения в отсутствии шума. Для этого мы будем использовать тот же самый алгоритм обратного распространения ошибки. Однако сам процесс обучения декомпозируем на 3 процедуры:

1. Обучение без шума ($n = 0$).

На этом этапе на вход нейронной сети подаются незашумлённые, чёткие символы. Обучаем сеть до тех пор, пока погрешность e не станет ниже 0.1.

2. Обучение с шумом ($n = 0.3$).

Теперь на вход нейронной сети подаём зашумлённые символы. Уровень шума выбираем 0.3. Обучаем сеть до тех пор, пока погрешность e не станет ниже 0.1.

3. Обучение без шума ($n = 0$).

Так как после обучения с шумом нейронная сеть могла «забыть», как выглядят чёткие символы, проведём повторное обучение сети без шума. Обучаем сеть до тех пор, пока погрешность e не станет ниже 0.1.

Проверку будем осуществлять на множестве, состоящем из 100 зашумлённых символов (уровень шума n задаётся заранее). Количество правильно распознанных символов даст нам процент распознавания при данном уровне шума.

Для оценки эффективности обучения в условиях шума проведите серию экспериментов:

Эксперимент №1. Обучите нейронную сеть без шума, используя только процедуру №1. Постройте зависимость процента распознавания символов от уровня зашумлённости входных данных.

Эксперимент №2. Обучите нейронную сеть последовательно с помощью процедур 1, 2, 3. В процедуре №2 уровень шума установите $\boldsymbol{n = 0.1}$. Постройте

зависимость процента распознавания символов от уровня зашумлённости входных данных.

Эксперимент №3. В отличие от эксперимента №2 установите в процедуре №2 уровень шума ***n* = 0.3**. Постройте зависимость процента распознавания символов от уровня зашумлённости входных данных.

Эксперимент №4. В отличие от эксперимента №2 установите в процедуре №2 уровень шума ***n* = 0.5**. Постройте зависимость процента распознавания символов от уровня зашумлённости входных данных.

Эксперимент №5. В отличие от эксперимента №2 установите в процедуре №2 уровень шума ***n* = 0.7**. Постройте зависимость процента распознавания символов от уровня зашумлённости входных данных.

Эксперимент №6. В отличие от эксперимента №2 установите в процедуре №2 уровень шума ***n* = 0.9**. Постройте зависимость процента распознавания символов от уровня зашумлённости входных данных.

Отчёт должен содержать:

1. Задание.
2. Ход выполнения работы.
3. Программный код или блок-схема алгоритма.
4. Графическое отображение результатов экспериментов. Желательно отобразить результаты экспериментов на одном графике.
5. Выводы об эффективности/неэффективности обучения нейронной сети в условиях шума.

Контрольные вопросы:

1. Схема многослойной нейронной сети для распознавания образов.
2. Описание алгоритма обратного распространения ошибки.
3. Формула корректировки весов в алгоритме обратного распространения.
4. Способы повышения сходимости процедуры обучения при использовании алгоритма обратного распространения.

Лабораторная работа №4
ОБУЧЕНИЕ МЫШКИ В Т–ЛАБИРИНТЕ

Цель работы: моделирование мозга мышки и её поведения в Т-образном лабиринте.

Теоретическая часть

Приведём подход, описанный В.Д. Цыганковым в [1, 2]; здесь и далее в лабораторных работах 5 и 6 – выдержки из книг автора подхода.

Сконструируем искусственный гипотетический мозг мышки, способный обучаться поведению в Т–образном лабиринте (рис. 4.4) путем многократного выбора из двух вероятностных альтернатив или направлений, путей движения. Каждый выбор направления заканчивается либо получением поощрения в виде пищи **q**, либо получением наказания в виде боли **p**.

Сначала создадим внутреннюю среду или внутреннюю память $\mathbf{P_л}$ мозга необученной мышки. Представим её в виде некоторого ящика, в который могут помещаться несколько черных шаров, число **q** которых эквивалентно количеству пищи, полученной слева в Т–лабиринте, и несколько белых шаров, число **p** которых эквивалентно количеству наказаний, полученной также слева. Следует иметь в виду, что общая сумма черных и белых шаров не может превысить размера ящика, максимальная вместимость которого равна **n** шаров, т.е. $\mathbf{p + q = n}$. Допустим, что имеется точно такая же память, с такими же свойствами $\mathbf{P_п}$ и для правой стороны нашего Т–лабиринта. В этих двух массивах памяти хранится прошлый опыт мышки о посещении Т–лабиринта слева и справа. Для конкретности допустим, что всего в левой или правой памяти $\mathbf{P_л}$ или $\mathbf{P_п}$ может быть не более и не менее 10 шаров, т. е. $\mathbf{p + q = 10}$. Соотношение цветов шаров в каждой памяти может меняться, но память всегда должна быть заполнена.

Теперь сконструируем в мозге мышки память о событиях во внешней среде в виде $\mathbf{S_л}$ и $\mathbf{S_п}$ массивов или ящиков с теми же свойствами, что и у массивов $\mathbf{P_л}$ и $\mathbf{P_п}$. Мы получили с Вами четыре массива памяти, по два для каждого

направления движения мышки в Т–лабиринте. Пусть, опять для конкретности, **p + q = 10**, однако, числа, входящие в сумму, уже будут относиться к состоянию внешней среды. У нас появилась внутренняя среда в виде пары массивов памяти $\{\mathbf{P}_л\}$ и $\{\mathbf{P}_п\}$ и внешняя среда, представленная в виде пары входных или сенсорных массивов памяти $\{\mathbf{S}_л\}$ и $\{\mathbf{S}_п\}$.

Числа **p** и **q** для каждой из сторон лабиринта представляют собою след в памяти о доли поощрений **q** или доли наказаний **p** в общем количестве посещений конкретной стороны лабиринта. Пусть, например, предыдущий опыт посещения мышкой лабиринта выглядит следующим образом: в левом $\mathbf{P}_л$ массиве памяти $\mathbf{p}_л = 7$, $\mathbf{q}_л = 3$, т. е. семь раз слева мышку наказывали, а три раза кормили. Тогда, логично предположить, что в правом массиве $\mathbf{P}_п$ памяти мышки должны быть обратные значения чисел наказания и поощрения, т.е. $\mathbf{p}_п = 3$, $\mathbf{q}_п = 7$. Это так называемый обычный случай симметричности внешних в Т–лабиринте воздействий слева и справа и их следов во внутренней памяти, или $\mathbf{p}_л = \mathbf{q}_п$, и $\mathbf{p}_п = \mathbf{q}_л$. Было бы для мышки жутко, если бы ее били слева и справа одинаково, да еще и кормили одинаково. Зная, как изменяется содержимое памяти слева, мы однозначно можем определить состояние массива памяти справа. Таким образом, одна левая половина внутренней памяти оказывается как бы полным инверсным дублером другой правой половины. Можно получать информацию о состоянии правой $\mathbf{P}_п$ памяти, сняв через схемы инверторов сигналы с левого $\mathbf{P}_л$ массива памяти. Таким образом, можно отказаться от использования одного из массивов внутренней памяти, пусть это будет $\mathbf{P}_п$, и *получить значительную экономию в оборудовании при практической реализации электронного мозга мышки*. Теперь будем обозначать массив внутренней памяти символом **P**, внешнюю память символом **S**, а характеристику её содержимого просто числами **p** и **q**.

Приступим к формированию структуры и функции обучения в мозге мышки. Если мышка не обучена, то след ее прошлого опыта хранится в массиве памяти **P** в виде двух случайных чисел **p** и **q** например, в двоичном коде. Пусть разряды нашего **n = 10**-ти разрядного регистра внутренней памяти случайным

образом установлены в **«0»**-е состояние и их число **p** будет равно **5**, тогда число **q** разрядов в **«1»**-м состоянии также будет равно **5**. Число разрядов регистра, находящихся в **«1»**-ом состоянии, эквивалентно числу поощрений или числу случаев получения мышкой пищи слева. Число **«0»**-ых разрядов регистра – это число наказаний слева. При этом мышка ожидает наиболее вероятное, однако, случайное будущее состояние внешней среды **S** слева с таким же свойством, с таким же соотношением чисел **p** и **q**.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**=n**
S	1	1	0	0	1	1	0	0	1	1	
P	1	0	1	0	1	0	1	0	1	0	

Рис. 4.1. Пространственное расположение внешней **S** и внутренней памяти **P** мозга мышки

Расположим массивы памяти **P** и **S**, состоящие из одинакового количества разрядов-секторов друг над другом (рис. 4.1). Закрепим друг за другом соответствующие разряды **S** и **P**. При последовательном, в порядке возрастания номера, сравнении секторов получим следующую последовательность состояний пар разрядов (S/P): (1/1).(1/0).(0/1).(0/0).(1/1).(1/0)(0/1).(0/0).(1/0).(1/1).

Определим модуль невязки **|J|** как общее число несовпадающих разрядов. Из рис. 4.1 видно, что не совпадают состояния в ячейках с номерами 2, 3, 6, 7 и 10. Таким образом, модуль невязки равен **|J| = 5** или неравновесие внутренней среды по отношению к состоянию внешней среды составляет 50%. Наша мышка необучена и у неё нет предпочтения в выборе той или иной стороны лабиринта. Величина модуля невязки определяет интенсивность или мотивационную силу поведенческого акта, а вектор невязки определяет конкретный вид поведения, целесообразность или разумную направленность поведенческой реакции.

Опишем алгоритм взаимодействия между векторами **S** и **P**.

В мозге мышки имеется источник внутренней активности – генератор шума. Этот источник осуществляет запуск движения мышки по лабиринту и, при достижении мышки разветвления лабиринта, выбирает одно из двух направлений поворота. Это происходит путём равновероятного выбора одного из 10-ти разрядов массива **P**. Выпадение **«1»**-го разряда соответствует ожиданию мышки получить слева пишу, тогда она повернёт налево. Выпадение **«0»**-го разряда будет соответствовать ожиданию мышки получить слева наказание, и она повернёт направо.

В нашем примере (рис. 4.1) **p** = **q**, значит у мышки нет предпочтения в выборе между правым и левым направлением поворота. При проходе мышкой лабиринта возможны 4 различных случая её взаимодействия с внешней средой:

1. Генератор шума выбирает 1-й разряд памяти **P**. Его значение равно 1, значит мышка ожидает получить слева пищу и поворачивает налево. Соответствующий разряд массива **S** также равен 1, что означает что ожидания мышки совпали с реальностью. В этом случае её поведение не изменится.

2. Генератор шума выбирает 4-й разряд памяти **P**. Его значение равно 0, значит мышка ожидает получить слева наказание и поворачивает направо, где, по её предположению, её должны накормить. Соответствующий разряд массива **S** также равен 0, что означает, что ожидания мышки совпали с реальностью, так как внешние условия симметричны, а значит, справа мышку должны покормить. В этом случае её поведение также не изменится.

3. Генератор шума выбирает 3-й разряд памяти **P**. Его значение равно 1, значит, мышка ожидает получить слева пищу и поворачивает налево. Однако, соответствующий разряд массива **S** равен 0, и мышка получает наказание. Подобное несоответствие ожидания и действительности откладывает отпечаток в памяти мышки, и соответствующий разряд памяти **P** станет равен 0. Таким образом, поведение мышки изменится. Теперь **p = 4**, **q = 6**, значит, мышка будет чаще выбирать правый поворот.

4. Генератор шума выбирает 2-й разряд памяти **P**. Его значение равно 0, значит, мышка ожидает получить слева наказание и поворачивает направо.

Однако, соответствующий разряд массива **S** равен 1, и, следовательно, справа мышка получает наказание. Несоответствие ожидания и действительности откладывается в памяти мышки, и соответствующий разряд памяти **P** станет равен 1. Поведение мышки изменится. Теперь **p = 6**, **q = 4**, значит, мышка будет чаще выбирать левый поворот.

При каждой новой пробежке разряды массива **S** случайным образом меняются местами, при этом должно сохраняться соотношение количества единичных и нулевых разрядов.

После многократного посещения мышкой лабиринта средний модуль невязки **|J|** уменьшится и мышка станет предпочитать тот поворот, где её чаще будут кормить. Мышка полностью обучится.

На рис. 4.2 представлена кривая обучения мозга мышки. По оси абсцисс отложены номера пробежек, а по оси ординат – вероятность наступления равновесия между **S** и **P**.

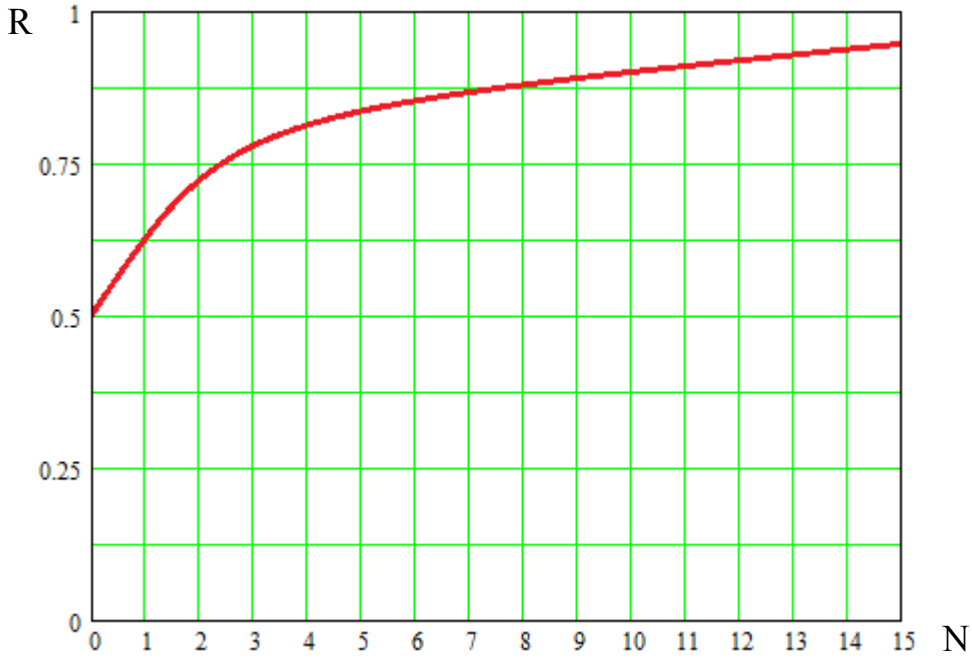


Рис. 4.2. Кривая обучения мышки в Т-лабиринте. R – вероятность наступления равновесия между **S** и **P**, N – число опытов или пробежек

На рис. 4.3 из книги Буша и Мостеллера [3] изображены опытные кривые обучения настоящих живых мышек. Как видно из сравнения графиков, наш

электронный мозг организует и демонстрирует поведение подобно живому мозгу в аналогичном опыте.

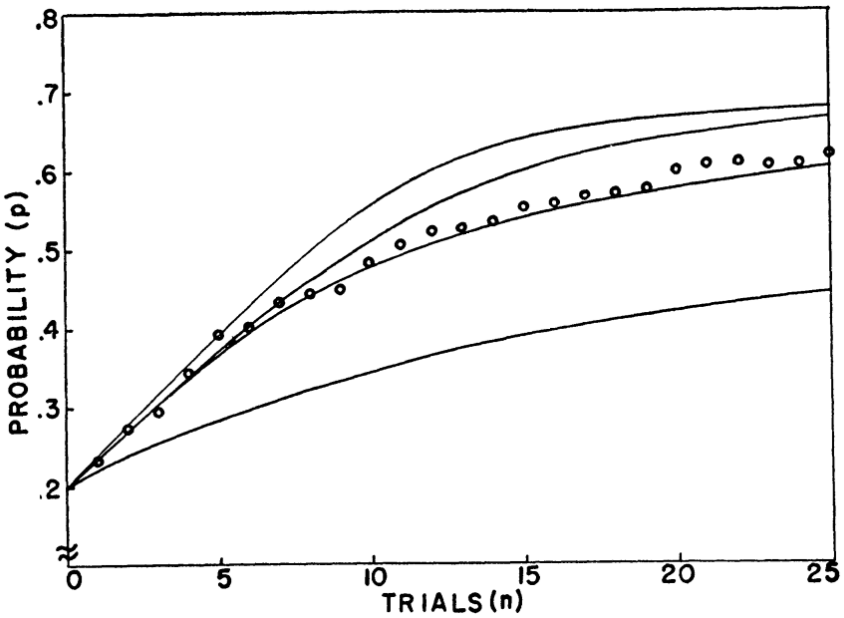


Рис. 4.3. Опытные кривые обучения живых мышек

Мозг мышки разумно управляет поведением мышки и адаптивно перестраивает его состояние памяти, постоянно отслеживая любые изменения во внешней среде и приспосабливая поведение к её статистически устойчивым свойствам.

Практическая часть

Задание: разработать алгоритм и программу, моделирующую мозг мышки и её поведение в Т-лабиринте.

Порядок выполнения работы:

Создайте массивы **S** и **P**, каждый из них пусть содержит по 10 элементов (см. рис. 4.1). Заполните массив **S** единицами и нулями. Соотношение между количествами нулей и единиц будет определять соотношение между наказанием и болью, полученными в левой стороне лабиринта. Массив **P** заполните нулями и единицами случайный образом.

Разработайте процедуру запуска мышки в Т-лабиринт и адаптации её мозга к условиям внешней среды (соотношение между **p** и **q**). При этом порядок расположения элементов массива **S** меняется случайным образом.

Посчитайте количество пробежек, требуемое для адаптации мозга мышки к условиям внешней среды. Промоделируйте 100 пробежек и посчитайте, сколько раз мышка повернула в ту или иную сторону и сколько раз её покормили.

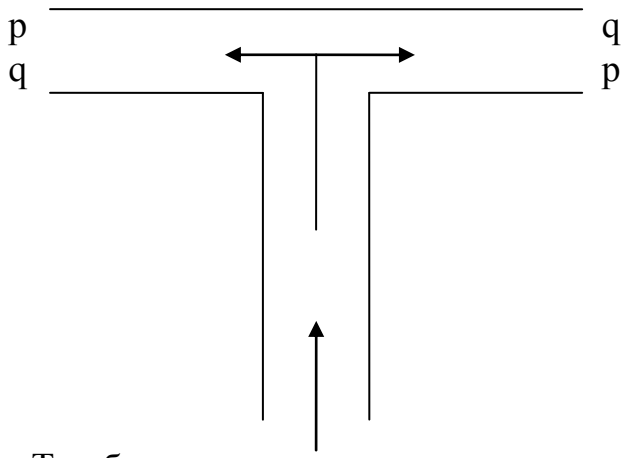


Рис. 4.4. Схема Т-лабиринта

Постройте кривую обучения мозга мышки, откладывая по оси абсцисс количество пробежек, а по оси ординат – степень её обученности (как на рис. 4.2).

Отчёт должен содержать:

1. Задание.
2. Ход выполнения работы.
3. Программный код или блок-схема алгоритма.
4. Результаты работы программы.
5. Графическое отображение процедуры обучения мышки.

Контрольные вопросы:

1. Описание опыта обучения мышек в Т-лабиринте.
2. Отображение внутренней и внешней памяти в электронном мозге.
3. Порядок взаимодействия элементов массивов S и P.

Литература:

1. Цыганков В.Д. Нейрокомпьютер и мозг, М.: СИНТЕГ, 2001, 248 с.

2. Цыганков В.Д. Виртуальный нейрокомпьютер «Эмбрион», М.: СИНТЕГ, 2005, 184 с.

3. Bush, R. & Mosteller, F. (1955). Stochastic Models of Learning. John Wiliey & Son, New York

Лабораторная работа №5
МОДЕЛИРОВАНИЕ ВИРТУАЛЬНЫХ НЕЙРОННЫХ СЕТЕЙ

Цель работы: моделирование виртуальных нейронных сетей в парадигме «Эмбрион». Расчёт теоретических вероятностей для нейронов *n*-го слоя на уровне кодов и на уровне групп.

Теоретическая часть

Рассмотрим информационный 2-атом, предназначенный для генерации конкретного вида виртуальной нейронной сети.

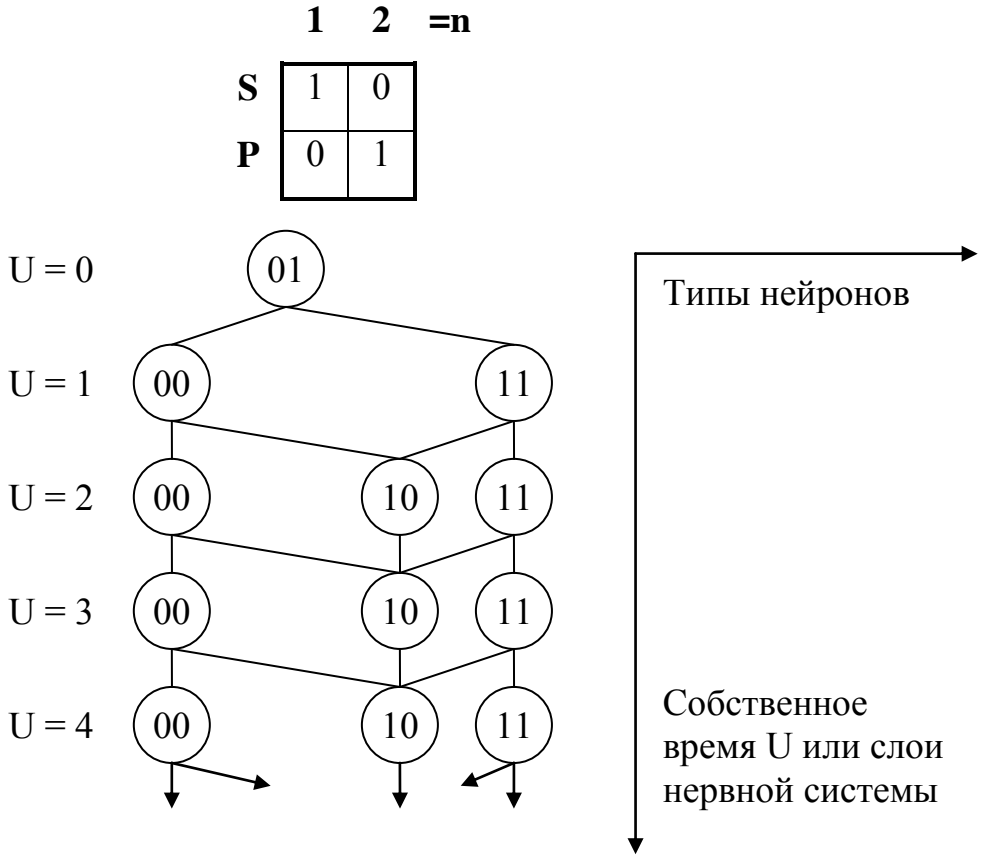


Рис. 5.1. 2-нейропроцессор и его виртуальная нейронная сеть

Рассмотрим подробно рис. 5.1. Будем сверху вниз, начиная с **U** = 1, горизонтальные линии считать слоями нейронной сети. Всего слоёв у нас 4. Кружки с двумя символами внутри на этих горизонталях будем называть квазинейронами.

Символы (01), (00), (11), (10) в кружках или двоичные коды состояния двухразрядного регистра **P** обозначают типы нейронов. Всего при **n** разрядах

возможное число состояний регистра составляет множество $\{\mathbf{X}\} = 2^n$ кодов. При 2-хразрядах будет всего 4 типа нейронов, а при 10 разрядах количество нейронов уже возрастёт до 1024.

В нулевом слое изображённой на рис. 5.1 сети или на оси нулевого момента (**U** = 0) собственного времени возбуждения сети находится скрытый нейрон, будем называть его «пейсмеккерным центром» или клеткой-«водителем внутреннего ритма активности». От неё возбуждается вся нейронная сеть.

Необходимо ввести в рассмотрение некоторые элементы в блоки нейропроцессора, которых нет на рис. 5.1. В нейрокомпьютере имеется главный импульсный генератор **NS**, который задаёт всю работу нейронной сети и все её внутренние и внешние ритмы. Так как наш нейрокомпьютер представляет собой вероятностный или стохастический автомат с переменной структурой и со случайными связями между элементами, то имеется внутри ещё один активный элемент – генератор шума, определяющий вероятности перехода между двумя элементами сети.

Связи между нейронами (01) – (00), (01) – (11) – это вероятности переходов, равные ½. Связи между нейронами (00) – (10), (00) – (00), (11) – (10), (11) – (11) – это вероятности переходов, также равные ½. Связь (10) – (10) – это вероятность перехода, равная 1.

Практическая часть

Задание: разработать программу, графически отображающую виртуальную нейронную сеть для **n** = 2, **U** = 4, произвольных значений **S** и **P**, и рассчитывающую теоретические вероятности нейронов на каждом шаге **U**.

Порядок выполнения работы:

Создайте массивы **S** и **P**, каждый из них будет состоять из двух бинарных элементов, например (11) и (00). Теперь на каждом шаге **U** генератор шума выбирает случайно один из разрядов. Соответствующий разряд матрицы **S** переходит в соответствующий разряд матрицы **P**. На 1-м шаге (**U** = 1) внутренняя память **P** может перейти из состояния (00) состояние (01) или (10).

Далее на 2-м шаге из состояния (01) возможен переход в состояние (01) или (11), а из состояния (10) – в состояние (10) или (11). Все вероятности переходов равны ½. Состояние (11) может перейти только в себя, поэтому вероятность такого перехода будет равна 1.

Исходя из вышесказанного, можно представить нейрокомпьютер как стохастический автомат Маркова (рис. 5.2).

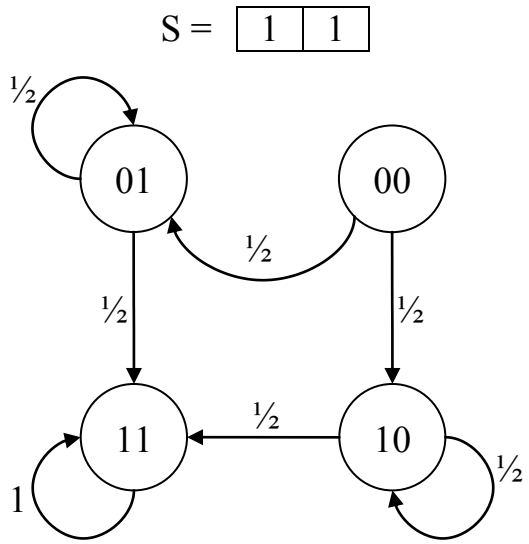


Рис. 5.2. Граф переходов для состояний внутренней памяти **P** для случая **S** = 11

На основании графа переходов составляем матрицу переходов:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ ½ & ½ & 0 & 0 \\ ½ & 0 & ½ & 0 \\ 0 & ½ & ½ & 1 \end{pmatrix}$$

Так как изначально автомат находится в состоянии (00), вектор вероятностей p_0 на нулевом шаге будет выглядеть так: $p_0 = (1 \quad 0 \quad 0 \quad 0)^T$.

Тогда согласно теории Марковских цепей

$$p_1 = Ap_0,$$

$$p_n = Ap_{n-1}, \text{ при } n \geq 1 \text{ или}$$

$$p_n = A^n p_0$$

Рассчитаем теоретические вероятности:

$$p_1 = \begin{pmatrix} 0 \\ 0.5 \\ 0.5 \\ 0 \end{pmatrix}, \quad p_2 = \begin{pmatrix} 0 \\ 0.25 \\ 0.25 \\ 0.5 \end{pmatrix}, \quad p_3 = \begin{pmatrix} 0 \\ 0.125 \\ 0.125 \\ 0.75 \end{pmatrix}, \quad p_4 = \begin{pmatrix} 0 \\ 0.063 \\ 0.063 \\ 0.875 \end{pmatrix}$$

На рис. 5.3 изображена виртуальная нейронная сеть с учётом посчитанных вероятностей. На нём отчётливо видно возбуждение нейронной сети при **U** = 1, а при **U** > 1 начинается процесс адаптации (как в опыте с мышкой), «энергия» возбуждения концентрируется в одном состоянии, соответствующему условиям внешней среды (значение **S**).

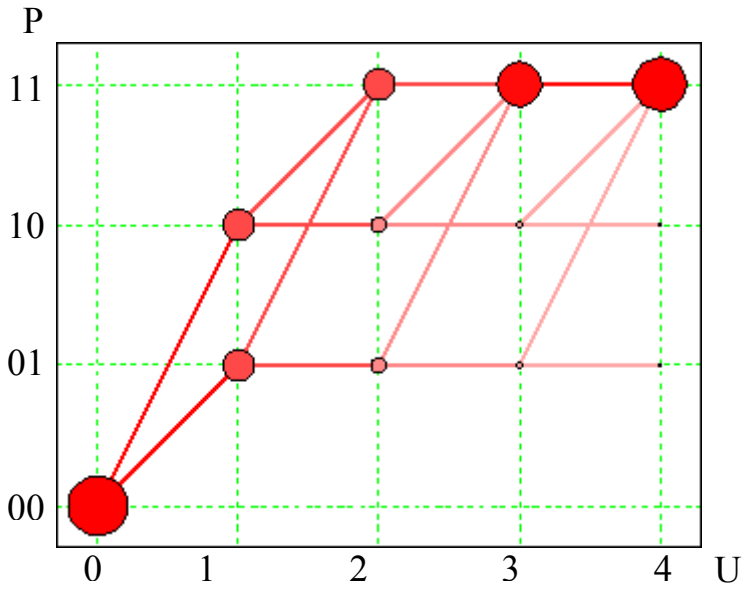


Рис. 5.3. Виртуальная нейронная сеть для **S** = 11, **P** = 00

Вероятности возбуждения нейронов прямо не влияют на изменение поведения адаптивной системы. За это отвечают импульсы *мотонейронов*, получающиеся путём сложения вероятностей нейронов в 3 группы. К первой группе относятся нейроны, коды которых не содержат ни одной единицы (для двухразрядной машины это единственный код 00). Ко второй группе относятся нейроны, в коде которых содержится только одна единица (01 и 10). К третьей группе относятся нейроны, в коде которых содержится две единицы (11).

Получаем 3-х канальный поток импульсов мотонейронов:

$$y_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad y_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad y_2 = \begin{pmatrix} 0 \\ 0.5 \\ 0.5 \end{pmatrix}, \quad y_3 = \begin{pmatrix} 0 \\ 0.25 \\ 0.75 \end{pmatrix}, \quad y_4 = \begin{pmatrix} 0 \\ 0.125 \\ 0.875 \end{pmatrix}$$

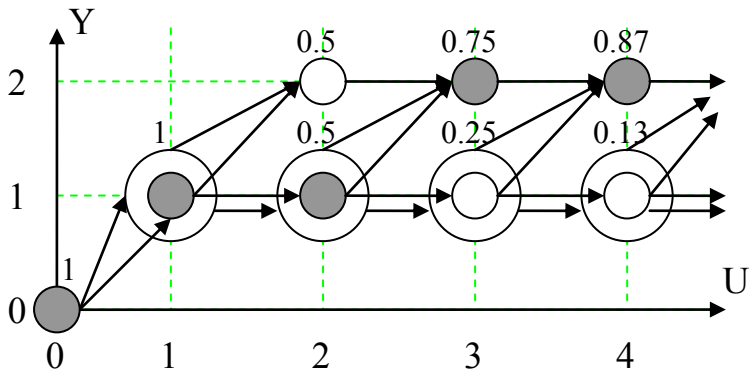


Рис. 5.4. Виртуальная нейронная сеть на уровне групп

Трёхканальный поток импульсов мотонейронов, изображённый на рис. 5.4 передаётся напрямую на *исполнительные органы*, в частности, на двигатели.

Отчёт должен содержать:

1. Задание.
2. Ход выполнения работы.
3. Программный код или блок-схема алгоритма.
4. Рисунки виртуальных нейронных сетей для случаев **S**/**P** = (01/10), (10/00), (11/11), **U** = 4.
5. Рассчитанные теоретические вероятности нейронов на уровне кодов и на уровне групп для каждого случая.

Контрольные вопросы:

1. Описание процедуры генерации виртуальной нейронной сети.
2. Расчёт теоретических вероятностей нейронов и мотонейронов.
3. Представление нейрокомпьютера как стохастического автомата Маркова.

Литература:

1. Цыганков В.Д. Нейрокомпьютер и мозг, М.: СИНТЕГ, 2001, 248 с.

2. Цыганков В.Д. Виртуальный нейрокомпьютер «Эмбрион», М.: СИНТЕГ, 2005, 184 с.

3. Цыганков В.Д. Квантовые вычисления на нейрокомпьютере «Эмбрион-10К», Москва, 2010, 176 с.

Лабораторная работа №6

РАСПОЗНАВАНИЕ ОБРАЗОВ СРЕДСТВАМИ НЕЙРОКОМПЬЮТЕРА

Цель работы: применение квантового нейрокомпьютера для решения задачи распознавания зрительных образов.

Теоретическая часть

Рассмотрим структуру и функционирование живого мозга, их связь со структурой и принципом работы нейрокомпьютера «Эмбрион».

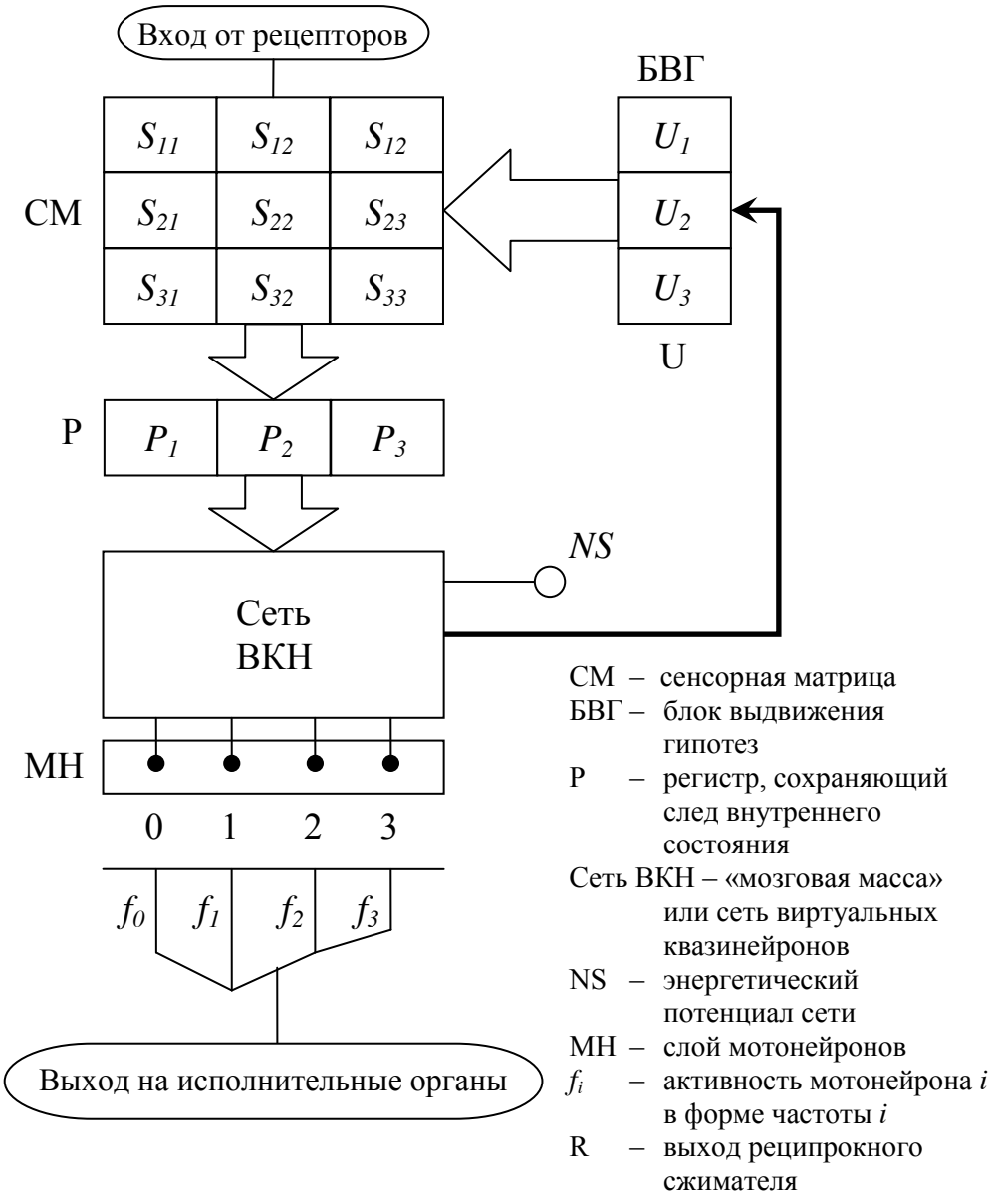


Рис. 6.1. Блок-схема нейрокомпьютера «Эмбрион»

Известно, что мозг насекомого, пресмыкающегося, рыбы, птицы, животного и человека – это средство выживания организма в незнакомой среде, средство координации поведения и двигательной активности.

На рис. 6.1 представлена блок-схема нейрокомпьютера, которая в своей основе достаточно полно повторяет структуру живого мозга и функционирует мозгоподобно.

Как и в настоящем мозгу, сигналы из внешней среды поступают в нейрокомпьютер через рецепторы – датчики или сенсоры входных воздействий. Здесь сигналы от специализированных групп рецепторов или первичных рецептивных полей перекодируются во внутренний код нервной системы, в многоканальный поток импульсов стандартной величины и длительности. Эта информация поступает, проецируется по принципу «точка в точку» на проекционное поле корковых нейронов – ячеек памяти в виде Сенсорной Матрицы (СМ) из n столбцов и m строк, где и хранится некоторое время в виде образа-мозаики или набора состояний «1» и «0» ячеек Памяти. Каждая строка СМ может рассматриваться как рецептивное поле второго уровня.

Информация из Сенсорной Матрицы построчно переносится в виде возбуждения в Регистр внутренней памяти (Р). Этот процесс переноса осуществляется под воздействием импульсного активирующего влияния подкоркового образования. Эту роль в живом мозгу играет ретикулярная формация, а в нашем нейрокомпьютере – это Блок Выдвижения Гипотез (БВГ). U_1, U_2, U_3 – это символы внутреннего времени. Они играют роль *гипотезы восприятия* или величины построчного *внимания*.

Этот процесс является причиной возникновения виртуальной ассоциативной нейронной сети – Сети Виртуальных Квазинейронов (Сеть ВКН), время возбуждения которой определяется энергетическим потенциалом сети, генератором активности NS – числом импульсов повторения или циклов возбуждения.

Многоканальные (2^n) потоки импульсов сжимаются, сворачиваются в виде нейрофизиологической «воронки Шеррингтона» на уровне спинного мозга в

компактный (n+1)-канальный поток импульсов возбуждения мотонейронов (МН). При 3-разрядном нейрокомпьютере это четыре канала 0, 1, 2, 3 импульсного воздействия на исполнительные органы в частности, на мышцы, на двигатели и т.д.

Выдающийся советский нейрофизиолог академик П.К. Анохин еще в 1935 году, задолго до кибернетики Н. Винера, открыл основополагающий «принцип обратной связи» в работе головного мозга, разработал и экспериментально обосновал теорию функциональных систем мозга.

Кратко, суть теории функциональной системы в следующем. Мозговая деятельность протекает в виде определенной архитектуры. Целью любой мозговой деятельности является *адаптация*, *обучение*, *приспособление*, *выживание*, выражающиеся в получении конкретного «полезного результата». Первая стадия мозгового процесса называется «афферентный синтез». На этой стадии сенсорная или афферентная (входная) информация из центра мотивации, от обстановочных раздражителей, информация из архивов генетической и прижизненной памяти, от внешнего или внутреннего пускового раздражителя анализируется, отбирается, синтезируется и ассоциативно объединяется. В результате формируется следующая стадия «принятие решения» на выполнение *конкретного* действия, на выполнение *конкретного* поведения в *конкретной* ситуации. В этот момент динамически формируются две нервные сети или структуры: «программа действия» и аппарат прогноза будущего полезного результата или «акцептор полезного результата действия». Следующий этап – это реализация программы действия и получение «полезного результата» или цели поведения, для достижения которой экстренно сформировалась в данный момент данная конкретная функциональная система, данная структура нервных связей и нервных возбуждений. В этот момент вступает в действие «акцептор полезного результата действия». Происходит сравнение параметров полученного «полезного результата» с параметрами ожидаемого результата, записанными в памяти «акцептора полезного результата действия». Если нет существенного

рассогласования, то «акцептор» санкционирует данное полезное действие, в противном случае срабатывает еще один контур обратной связи от результата через «акцептор» к «афферентному синтезу», что реализуется в виде «ориентировочного рефлекса». Система ищет новую информацию и вновь начинается «афферентный синтез». Если совпадение ожидаемого и фактически достигнутого результата налицо, то функциональная система выполнила свою роль, она тут же разрушается, и экстренно создается новая функциональная система.

Практическая часть

Задание: разработать программу для распознавания зрительных образов посредством нейросетевой парадигмы «Эмбрион».

Порядок выполнения работы:

В качестве символов, которые мы будем распознавать, воспользуемся уже готовым файлом, созданным для лабораторных работ № 2 и 3 (рис 2.2). Сенсорная матрица нейрокомпьютера **СМ** в таком случае будет иметь 5 строк и 4 столбца. Вектор состояния внутренней памяти **Р** будет иметь n = 4 разряда, и количество различных нейронов в каждом слое возбуждаемой сети будет равно 2^4=16. Количество различных мотонейронов будет равно n + 1 = 5.

Для каждой из пяти строк сенсорной матрицы задаём величину внимания U_i, i = 1, 2 ,…, 5. Для конкретности примем $U_1 = U_2 = \ldots = U_5 = 3$.

Как вы уже знаете из двух предыдущих лабораторных работ, моделирование мозговой деятельности реализуется довольно просто. Однако тогда мы не касались вопросов представления долговременной памяти, которая нужна нам для запоминания символов. Рассмотрим способ запоминания какого-либо объекта, образ которого попадает на сенсорную матрицу.

Как и в предыдущей лабораторной работе, необходимо составить матрицу переходов, которая будет состоять из 16 столбцов и 16 строк. Всего в матрице будет 256 элементов, поэтому процесс её заполнения нужно автоматизировать. Такая матрица задаётся для каждой строки сенсорной матрицы (A_i, i = 1,2,…,5).

Приведём порядок расчёта теоретических вероятностей:

1. Подаём на сенсорную матрицу очередной символ. Будем считывать его построчно сверху вниз.

2. Задаём случайным образом начальное состояние внутренней памяти **P**. Соответствующий элемент вектора вероятностей p_0 будет равен 1. Остальные элементы обнуляем.

3. Для первой строки сенсорной матрицы определяем матрицу переходов A_1. Вычисляем:

$$p_1 = p_0A_1,\ p_2 = p_1A_1,\ p_3 = p_2A_1.$$

Далее для второй строки сенсорной матрицы определяем матрицу переходов A_2. Вычисляем:

$$p_4 = p_3A_2,\ p_5 = p_4A_2,\ \mathrm{p}_6 = p_5A_2.$$

и т. д. проделываем то же самое для остальных трёх строк сенсорной матрицы до получения вектора p_{15}.

4. Пункты 1-3 проделываем для всех символов из обучающего множества.

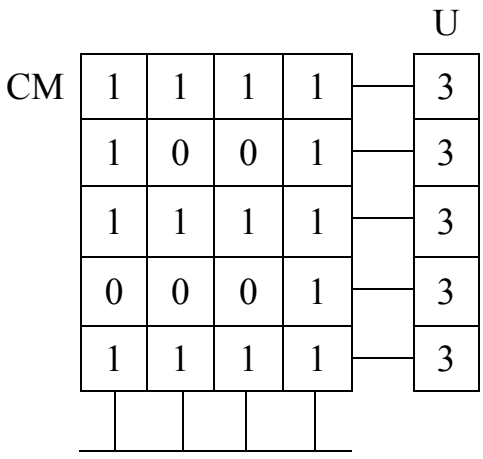
Так, для цифры «9» вектор теоретических вероятностей p_{15} будет выглядеть следующим образом:

$p_{15} = (0\ \ 0.004\ \ 0\ \ 0.0466\ \ 0\ \ 0.044\ \ 0\ \ 0.163\ \ 0\ \ 0.053\ \ 0\ \ 0.197\ \ 0\ \ 0.187\ \ 0\ \ 0.305)$

Для более эффективного распознавания найдём вектор вероятностей на уровне групп. Вероятность каждой группы соответствует сумме вероятностей кодов, содержащих одинаковое число единиц (см. работу №5):

$$t_{15} = (0\ \ 0.004\ \ 0.144\ \ 0.547\ \ 0.305)$$

Для каждого символа из обучающего множества запоминаем соответствующий ему вектор t_{15}.



P	0	0	1	1

Рис 6.2. Схема взаимодействия блоков нейрокомпьютера «Эмбрион»

Распознавание подаваемого не сенсорную матрицу символа будет осуществляться путём сравнения уровней возбуждения на уровне групп вновь пришедшего символа со всеми запомненными.

Подадим на сенсорную матрицу цифру 9 (рис. 6.2). Значения элементов регистра P задаём произвольным образом.

Процесс возбуждения нейронной сети будет соответствовать многократным пробежкам мышки в Т-лабиринте. Для каждой строки сенсорной матрицы выбираются последовательно 3 случайных разряда и их значения переносятся в соответствующие разряды регистра **P**. Конечное состояние регистра **P** определяет, какой из 16-ти нейронов последнего слоя будет возбуждён.

Проделаем 1000 циклов возбуждения нейронной сети (*NS* = 100), для этого будем каждыё раз возвращать значение регистра **P** в исходное состояние (так как в живом мозге циклы возбуждения происходят параллельно, а не последовательно). Подсчитаем, сколько раз в конце каждого цикла возбуждения регистр **P** принимал те или иные значения. Для цифры «9» мы можем получить следующие частоты состояний (для удобства запишем в виде вектора *F*):

$$F = (0\ \ 1\ \ 0\ \ 43\ \ 0\ \ 38\ \ 0\ \ 170\ \ 0\ \ 59\ \ 0\ \ 198\ \ 0\ \ 178\ \ 0\ \ 313)$$

Если разделить этот вектор на NS, мы получим распределение вероятностей состояний

$$P = \frac{1}{NS} F = (0\ \ 0.001\ \ 0\ \ 0.043\ \ 0\ \ 0.038\ \ 0\ \ 0.17\ \ 0\ \ 0.059\ \ 0\ \ 0.198\ \ 0\ \ 0.178\ \ 0\ \ 0.313)$$

Находим вектор распределения вероятностей на уровне групп:

$$T = (0\ \ 0.001\ \ 0.140\ \ 0.546\ \ 0.313)$$

Сравним теперь вектор T с теоретически рассчитанными вероятностями для каждой цифры по методу наименьших квадратов и определим таким образом ответ нейрокомпьютера.

Эффективность подобного способа распознавания можно показать, вводя на сенсорную матрицу нейрокомпьютера искажённые символы, такие, как, например, на рис. 6.3.

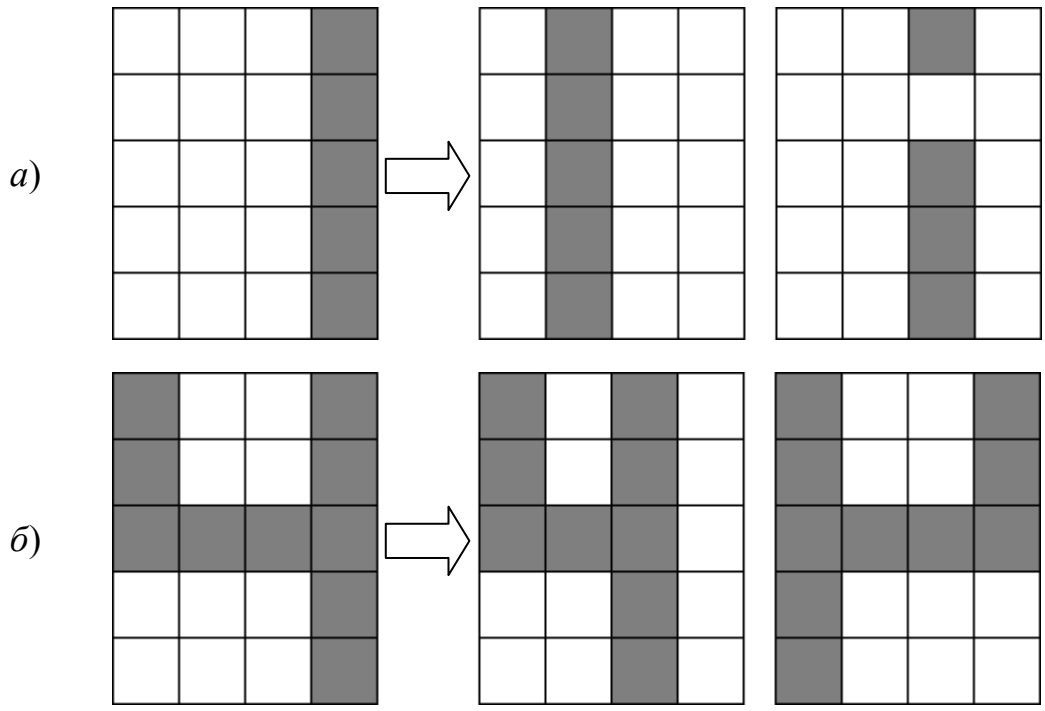


Рис. 6.3. Искажённые символы: *а*) цифра «1»; *б*) цифра «4»

Сравнивая результаты распознавания символов классическими нейронными сетями и посредством нейросетевой парадигмы «Эмбрион», можно заключить, что второй подход очень эффективно справляется с задачей распознавания сдвинутых и зеркально отражённых символов, что позволяет эффективнее использовать ресурсы аппаратных средств при реализации данной задачи.

Отчёт должен содержать:

1. Задание.
2. Ход выполнения работы.
3. Программный код или блок-схема алгоритма.
4. Примеры правильной и неправильной работы программы.
5. Примеры устранения неправильного распознавания символов.

Контрольные вопросы:

1. Схема функционирования живого мозга и нейрокомпьютера.
2. Блок-схема нейрокомпьютера «Эмбрион».

Литература:

1. Цыганков В.Д. Нейрокомпьютер и мозг, М.: СИНТЕГ, 2001, 248 с.

2. Цыганков В.Д. Виртуальный нейрокомпьютер «Эмбрион», М.: СИНТЕГ, 2005, 184 с.

3. Цыганков В.Д. Квантовые вычисления на нейрокомпьютере «Эмбрион-10К», Москва, 2010, 176 с.

4. Анохин П.К. Теория функциональной системы. // ж-л. Успехи физиологических наук, Том 1, № 1, 1970, с. 26–27.

5. Сеченов И.М. Рефлексы головного мозга. М.: АМН СССР, 1952. 186 с.

Лабораторная работа №7
ПОСТРОЕНИЕ НЕЧЁТКОЙ ЭКСПЕРТНОЙ СИСТЕМЫ

Цель работы: разработка системы нечёткого вывода для определения стоимости туристической поездки.

Теоретическая часть

Поставленная задача является одной из наиболее простых, которая может быть решена методами нечеткого моделирования. В качестве алгоритма нечеткого вывода будет использоваться алгоритм Мамдани.

Жизненный опыт позволяет сформулировать несколько эвристических правил, которые мы применяем для определения стоимости поездки:

1. Если едем в Европу или в плохую гостинцу, то цена будет низкой.

2. Если едем в Азию или в среднюю гостинцу, то цена будет средней.

3. Если едем в Америку или в гостиницу высшей категории, то цена будет высокой.

4. Если едем поездом, то цена будет низкой.

5. Если летим самолётом, то цена будет высокой.

Эта информация будет использоваться при построении базы правил системы нечеткого вывода, которая позволяет реализовать данную модель нечеткого управления.

Эта информация будет использоваться при построении базы правил системы нечеткого вывода, которая позволяет реализовать данную модель нечеткого управления.

Построение базы нечетких лингвистических правил

Для формирования базы правил систем нечеткого вывода необходимо предварительно определить входные и выходные лингвистические переменные. Очевидно, в качестве входных лингвистических переменных следует использовать дальность расстояния до места отдыха, класс гостиницы и вид транспорта, или формально: α_1 – «расстояние», α_2 – «класс гостиницы», α_3 –

«вид транспорта». В качестве выходной лингвистической переменной будем использовать стоимость поездки или формально: β_1 – «цена».

В этом случае система нечеткого вывода будет содержать 5 правил нечетких продукций следующего вида:

ПРАВИЛО 1: ЕСЛИ «Европа» ИЛИ «плохая гостиница» ТО «цена низкая»

ПРАВИЛО 2: ЕСЛИ «Азия» ИЛИ «средняя гостиница» ТО «цена средняя»

ПРАВИЛО 3: ЕСЛИ «Америка» ИЛИ «гостиница высшей категории» ТО «цена высокая»

ПРАВИЛО 4: ЕСЛИ «поезд» ТО «цена низкая»

ПРАВИЛО 5: ЕСЛИ «самолёт» ТО «цена высокая»

Фаззификация входных переменных

В качестве терм-множества первой лингвистической переменной будем использовать множество $T_1 = \{«Европа», «Азия», «Америка»\}$ с гауссовыми функциями принадлежности, изображенными на рис. 7.1*а*.

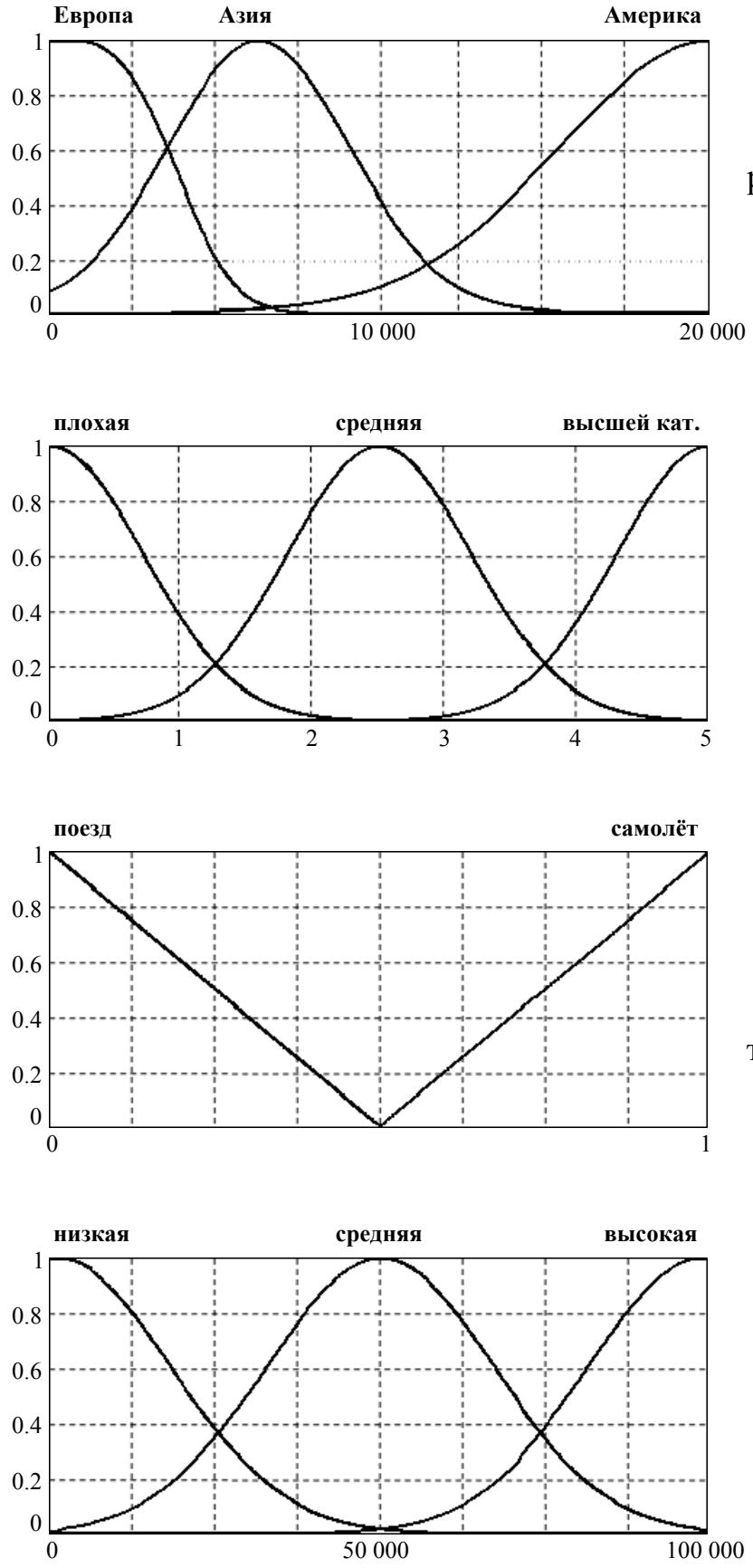
В качестве терм-множества второй лингвистической переменной будем использовать множество $T_2 = \{«плохая\ гостиница», «средняя\ гостиница», «гостиница\ высшей\ категории»\}$ также с гауссовыми функциями принадлежности, изображенными на рис. 7.1*б*.

В качестве терм-множества третьей лингвистической переменной будем использовать множество $T_3 = \{«поезд», «самолёт»\}$ с кусочно-линейными функциями принадлежности, изображенными на рис. 7.1*в*.

В качестве терм-множества четвёртой лингвистической переменной будем использовать множество $T_4 = \{«низкая\ цена», «средняя\ цена», «высокая\ цена»\}$ с гауссовыми функциями принадлежности, изображенными на рис. 7.1*г*.

При этом дальность поездки измеряется в километрах (от Москвы), класс гостиницы – по количеству звёзд, вид транспорта будет величиной безразмерной, а стоимость – в рублях.

Используя в качестве алгоритма вывода алгоритм Мамдани, рассмотрим пример его выполнения для случая, когда дальность поездки равна 2500 км, вид транспорта – самолёт (т.е. 1), класс гостиницы – 3 звезды.



а) расстояние (км)

б) гостиница (звёзды)

в) вид транспорта

г) цена (рубли)

Рис. 7.1. Функции принадлежности для термов лингвистических переменных

В этом случае фаззификация первой входной лингвистической переменной (расстояние) приводит к значениям степеней истинности ≈ 0.85 (Европа) для правила 1 и 0.4 (Азия) для правила 2. Эти значения определяют уровень отсечения для продукций соответствующих правил (рис. 7.2*а*).

Для второй входной переменной (класс гостиницы) значение истинности правила 2 составляет ≈ 0.8 (средняя гостиница), а для правила 3 – ≈ 0.05 (гостиница высшей категории). Как и в предыдущем случае, производим отсечение (рис. 7.2*б*).

Третья входная переменная (вид транспорта) даёт значение истинности, равное 1 (самолёт) для правила 5. Значит, для продукции соответствующего правила отсечение не производим (рис.7.2*в*).

Далее объединяем получившиеся функции принадлежности термов выходной лингвистической переменной (рис. 7.3).

Для дефаззификации выходной лингвистической переменной используем методом центра тяжести, что приводит к значению выходной переменной 51 000 руб. Это значение и является результатом решения задачи нечеткого вывода для текущих значений входных лингвистических переменных.

Практическая часть

Задание: придумать задачу, подобную рассмотренной в теоретической части, и разработать для неё систему нечёткого вывода с использованием алгоритма Мамдани.

Порядок выполнения работы:

Разработка системы нечёткого вывода может быть разбита на следующие этапы:

- Формирование базы правил систем нечеткого вывода. Правила устанавливаются исходя из результатов обработки статистических данных, опроса экспертов, жизненного опыта и т.п.
- Фаззификация входных переменных.

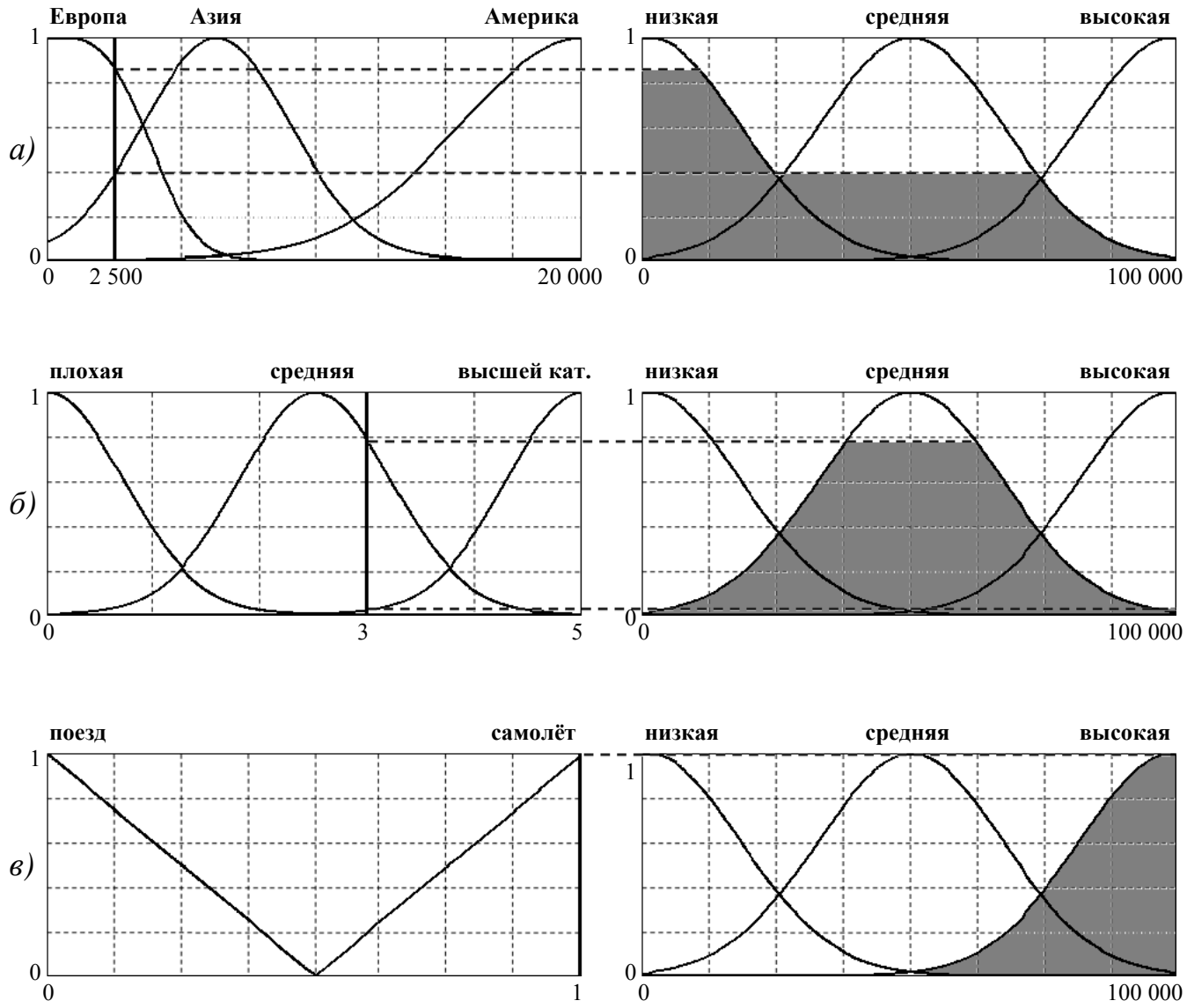


Рис. 7.2. Уровни отсечения для каждой из входных лингвистических переменных

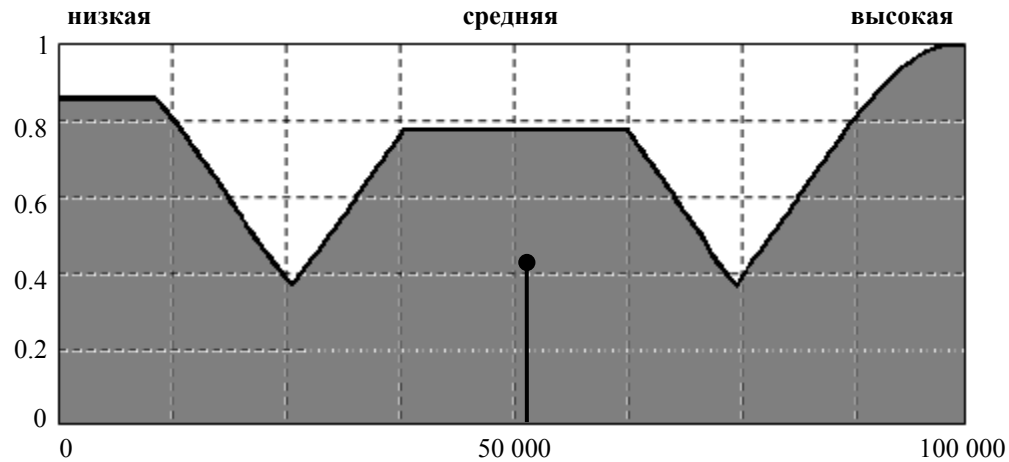


Рис. 7.3. Объединение функций принадлежности выходной переменной после аккумуляции

В качестве функций принадлежности можно использовать кусочно-линейные (рис. 7.1*в*), однако по вполне понятным соображениям наиболее популярны гауссовы функции.

Для функций принадлежности из рис. 7.1а были использованы следующие зависимости: $f(x) = e^{-(0.5x)^3}$, $f(x) = e^{-(0.5(x-3))^2}$, $f(x) = e^{-(0.3(x-10))^2}$.

- Агрегирование – процедура определения степени истинности условий по каждому из правил. Те правила, степень истинности условий которых отлична от нуля, считаются активными и используются для дальнейших расчетов.
- Активизация подзаключений в нечетких правилах продукций: представляет собой процедуру нахождения степени истинности каждого из подзаключений правил нечеткой продукции.
- Аккумуляция заключений нечетких правил продукций – процедура нахождения функции принадлежности для каждой из выходных лингвистических переменных.
- Дефаззификация выходных переменных. Традиционно используется метод центра тяжести или метод центра площади.

Центр тяжести или центроид площади рассчитывается по формуле:

$$y = \frac{\int_{Min}^{Max} x \cdot \mu(x)\,dx}{\int_{Min}^{Max} \mu(x)\,dx},$$

где y - результат дефаззификации; x - переменная, соответствующая выходной лингвистической переменной; $\mu(x)$ - функция принадлежности нечеткого множества, соответствующего выходной переменной после этапа аккумуляции; *Min* и *Max* - левая и правая точки интервала носителя нечеткого множества рассматриваемой выходной переменной.

Центр площади равен $y = u$, где значение u определяется из уравнения:

$$\int_{Min}^{u} \mu(x)\,dx = \int_{u}^{Max} \mu(x)\,dx .$$

Отчёт должен содержать:

1. Задание.
2. Подробное описание разрабатываемой системы нечёткого вывода
3. Ход выполнения работы.
4. Блок-схема алгоритма Мамдани.
5. Примеры работы программы для конкретных значений лингвистических переменных.

Контрольные вопросы:

1. Функции принадлежности.
2. Виды функций принадлежности.
3. Лингвистическая переменная
4. Нечёткий логический вывод. Алгоритм Мамдани

Литература:

1. Леоненков А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. – СПб.: БХВ Петербург, 2005, 736 с.

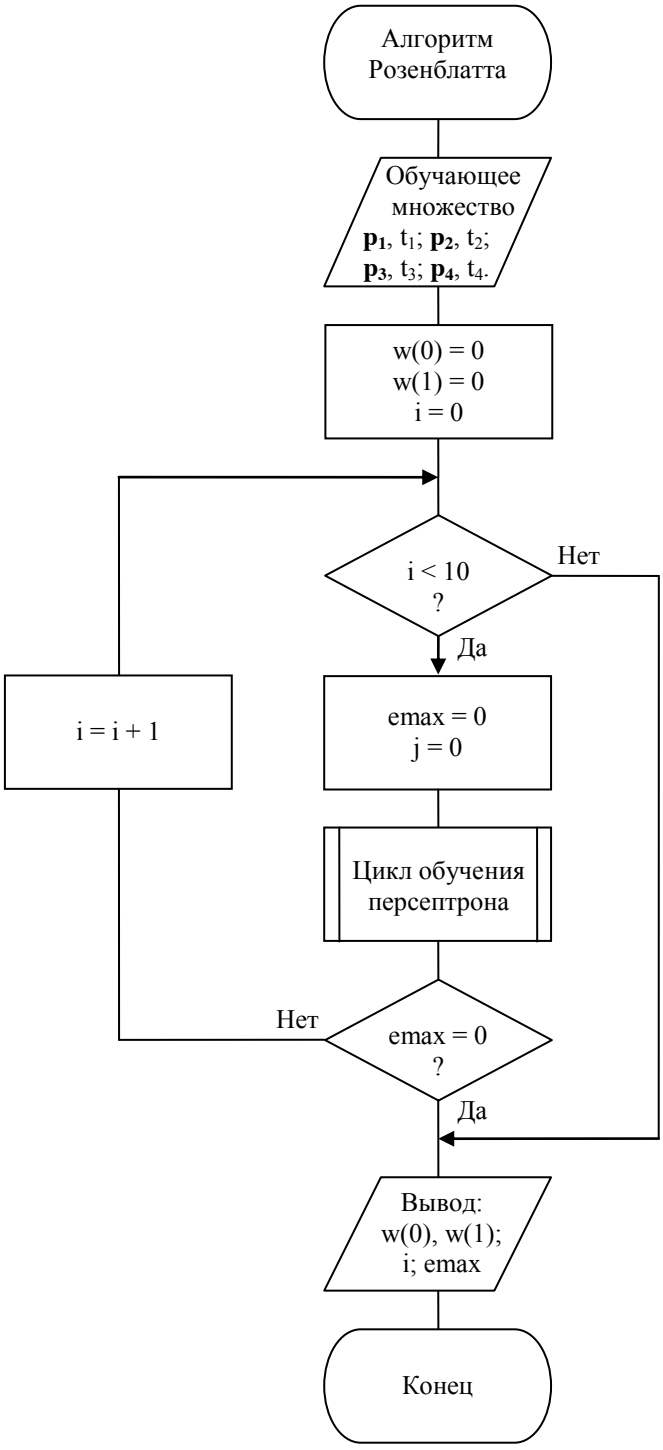
2. Круглов В.В., Дли М.И., Голунов Р.Ю. Нечёткая логика и искусственные нейронные сети. М.: ФИЗМАТЛИТ, 2001, 224 с.

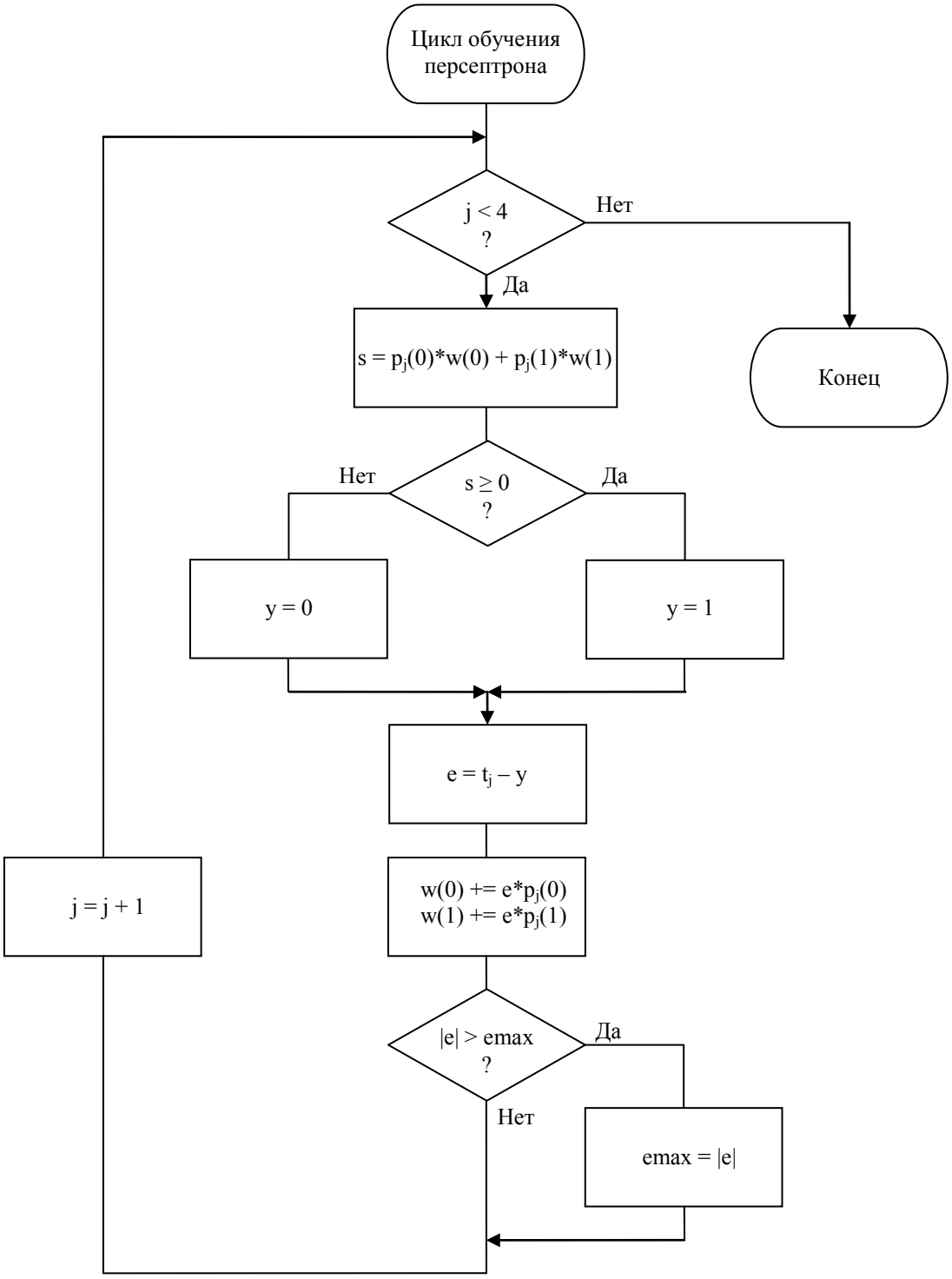
3. Степанян И.В. Интеллектуальные информационные системы. Лабораторный практикум. М.: МГГУ, 2007, 54 с.

Приложения

Приложение 1

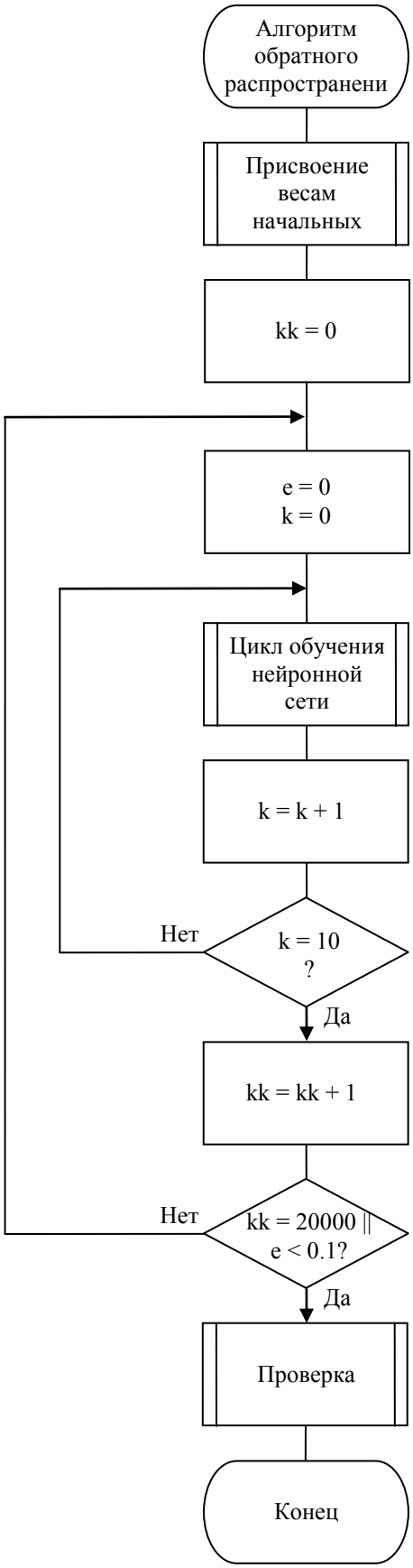
Блок-схема алгоритма для Лабораторной работы №1

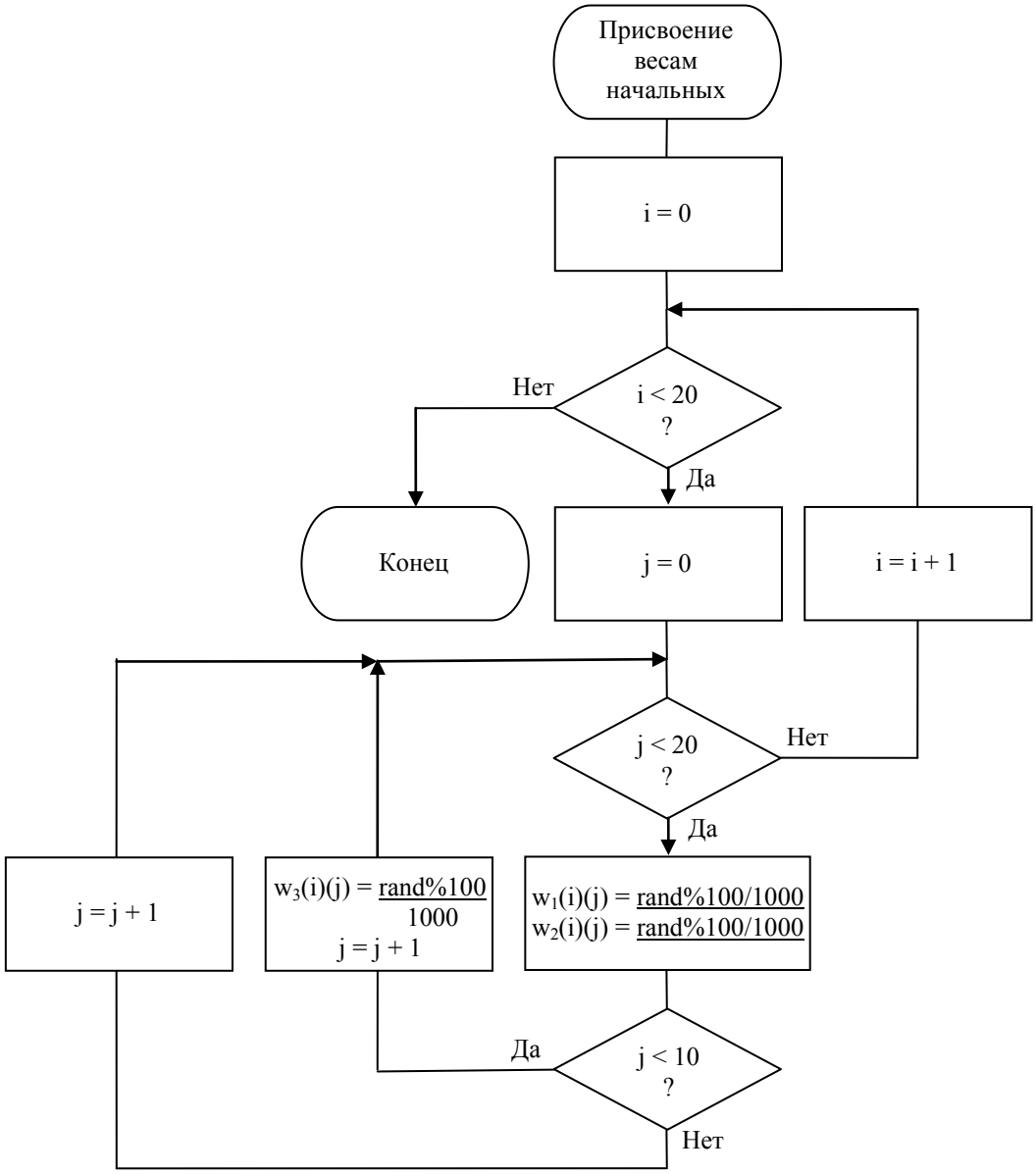



Цикл обучения персептрона
j < 4 ?
Нет
Да
Конец
s = p_j(0)*w(0) + p_j(1)*w(1)
s ≥ 0 ?
Нет
Да
y = 0
y = 1
e = t_j – y
j = j + 1
w(0) += e*p_j(0)
w(1) += e*p_j(1)
|e| > emax ?
Да
Нет
emax = |e|

Приложение 2

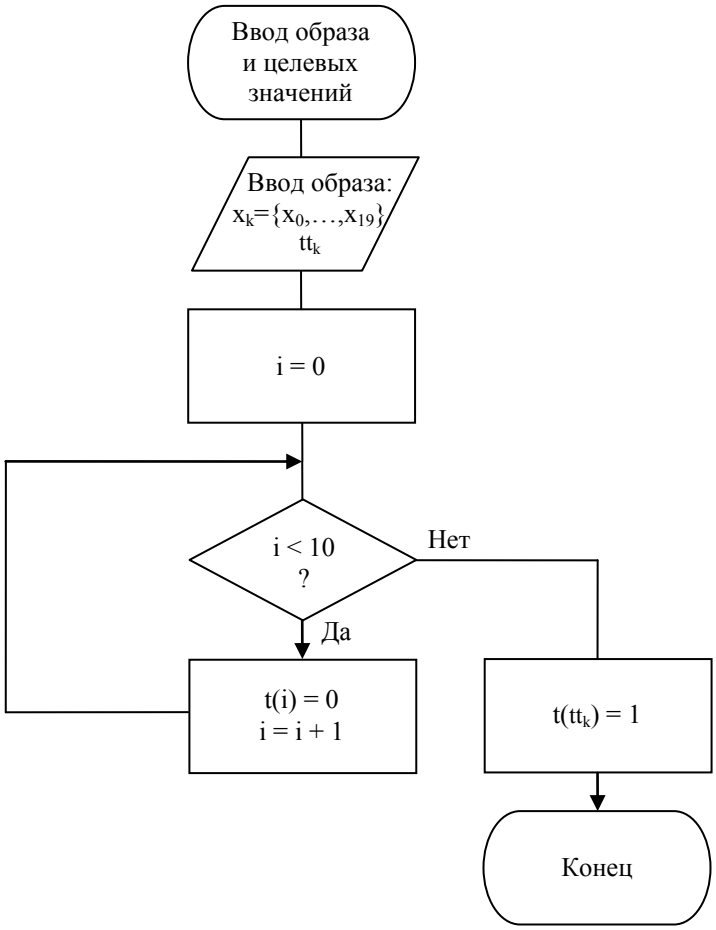
Блок-схема алгоритма для Лабораторной работы №2

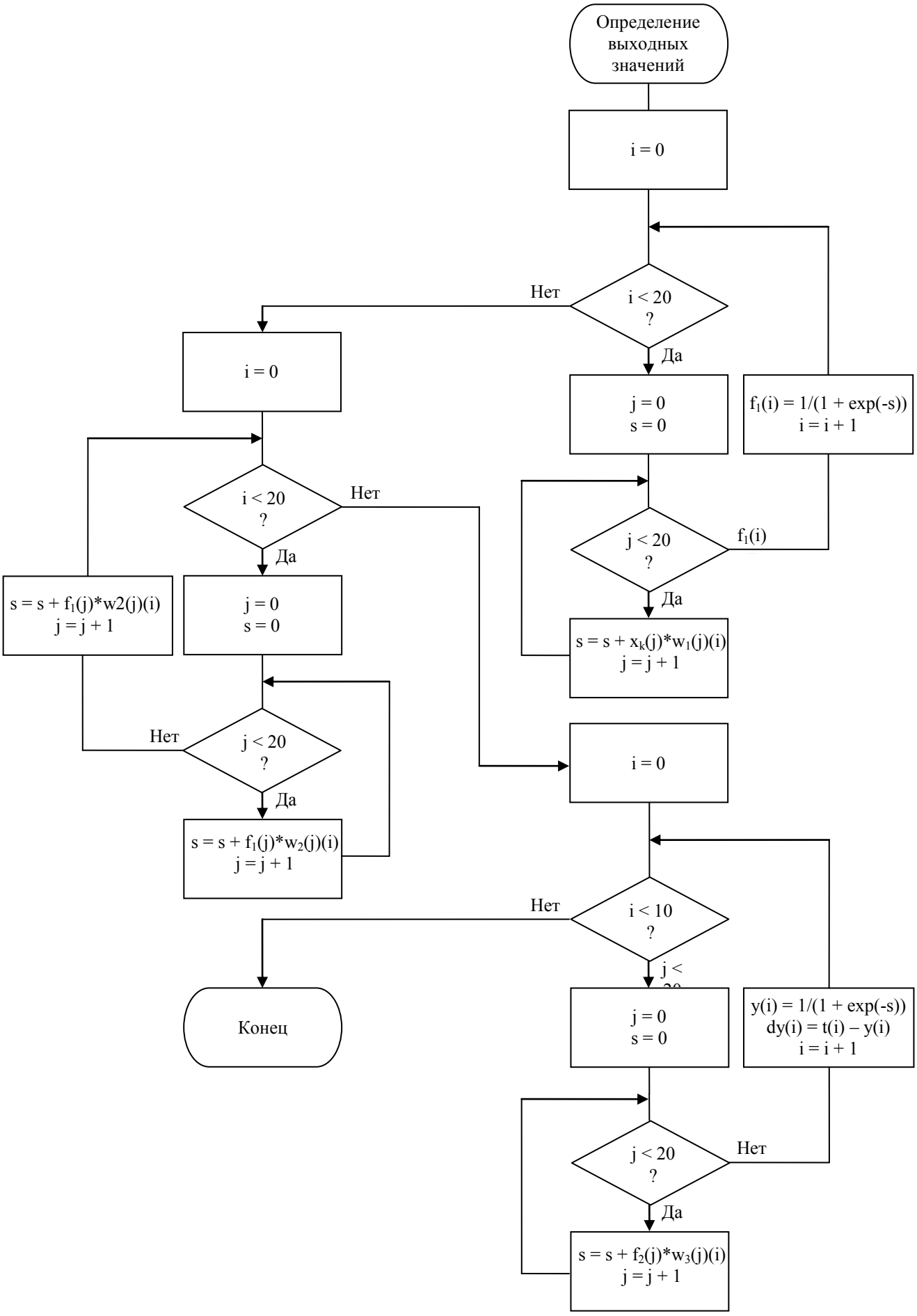


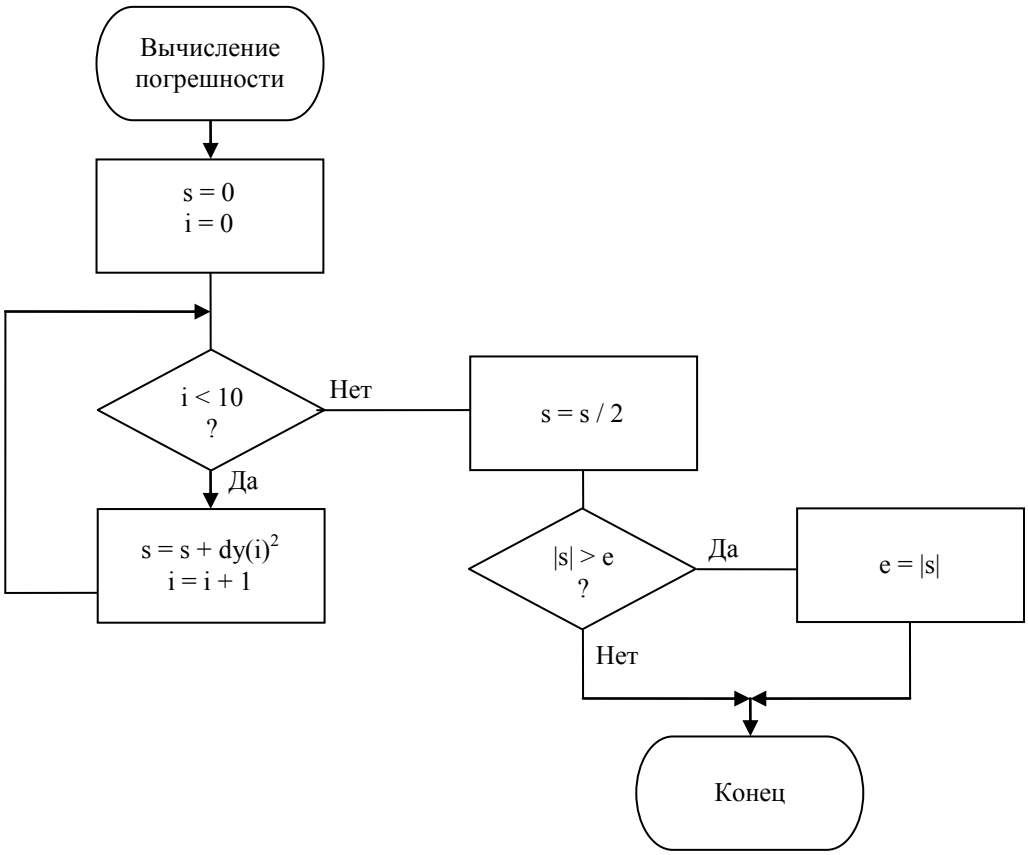

Присвоение весам начальных
i = 0
i < 20 ?
Нет
Да
Конец
j = 0
i = i + 1
j < 20 ?
Нет
Да
j = j + 1
w3(i)(j) = rand%100 / 1000
j = j + 1
w1(i)(j) = rand%100/1000
w2(i)(j) = rand%100/1000
Да
j < 10 ?
Нет

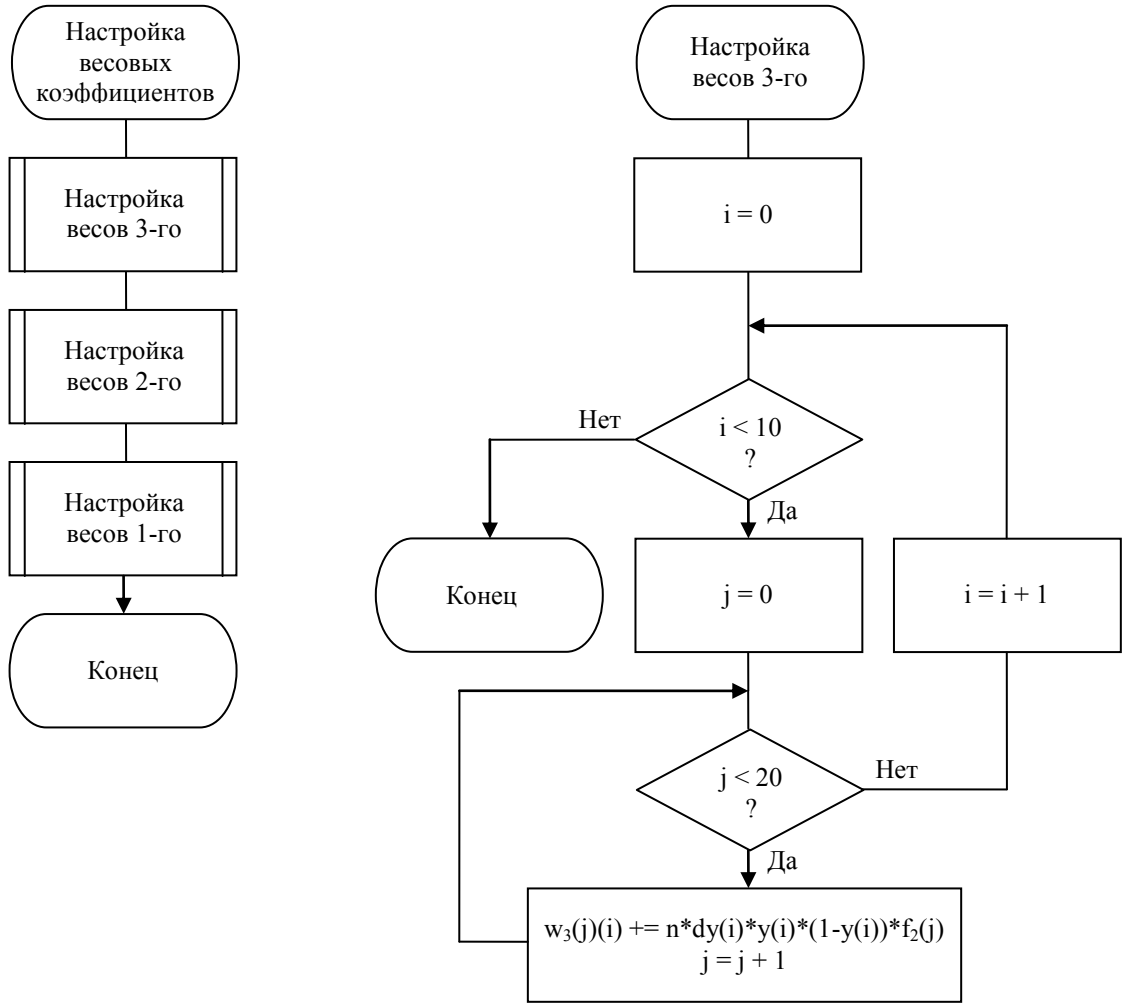

Цикл обучения нейронной сети
Вычисление выхода нейронной сети
Вычисление погрешности
Настройка весовых коэффициентов
Конец

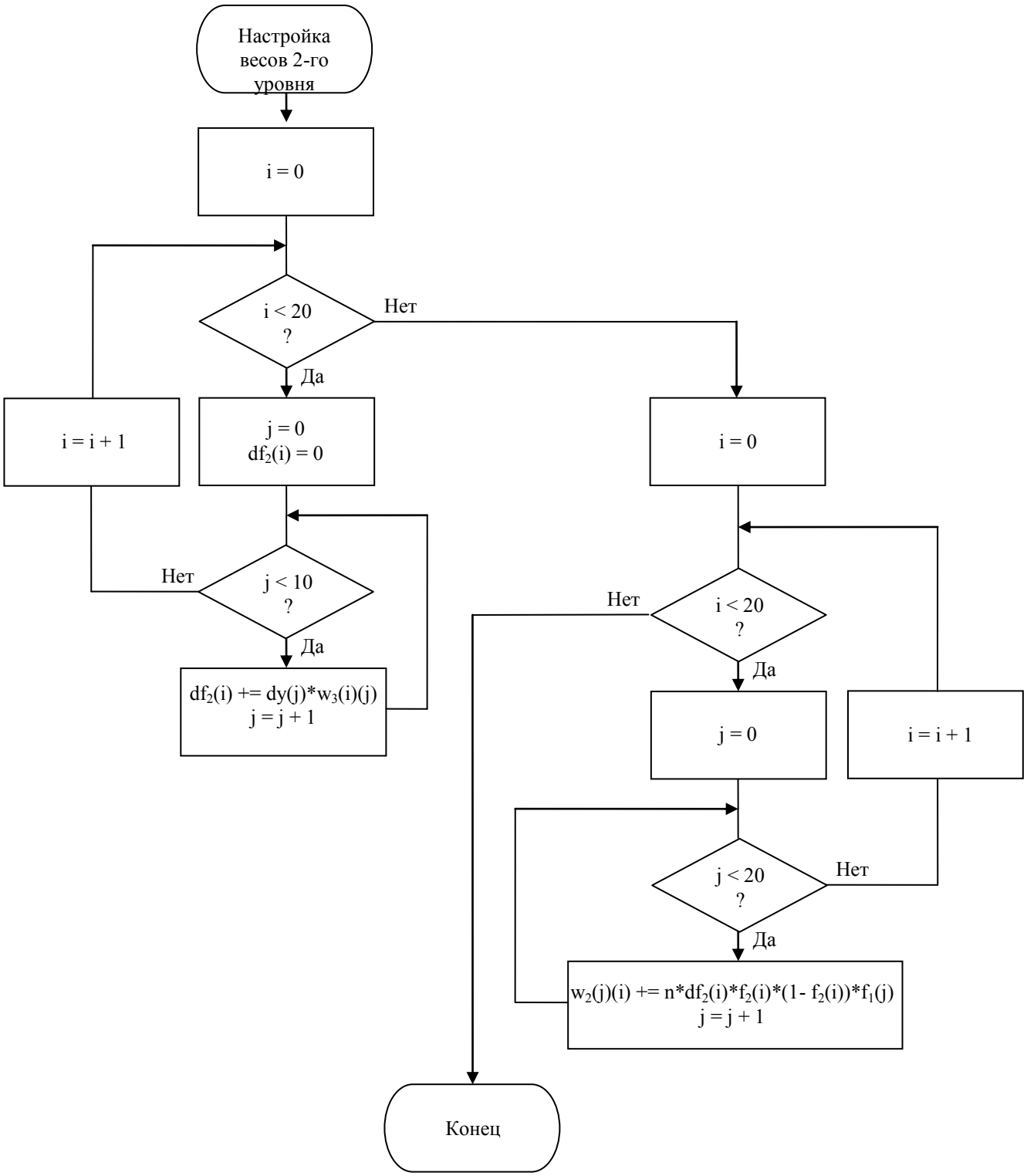

Вычисление выхода нейронной сети
Ввод образа и целевых значений
Определение выходных значений
Конец

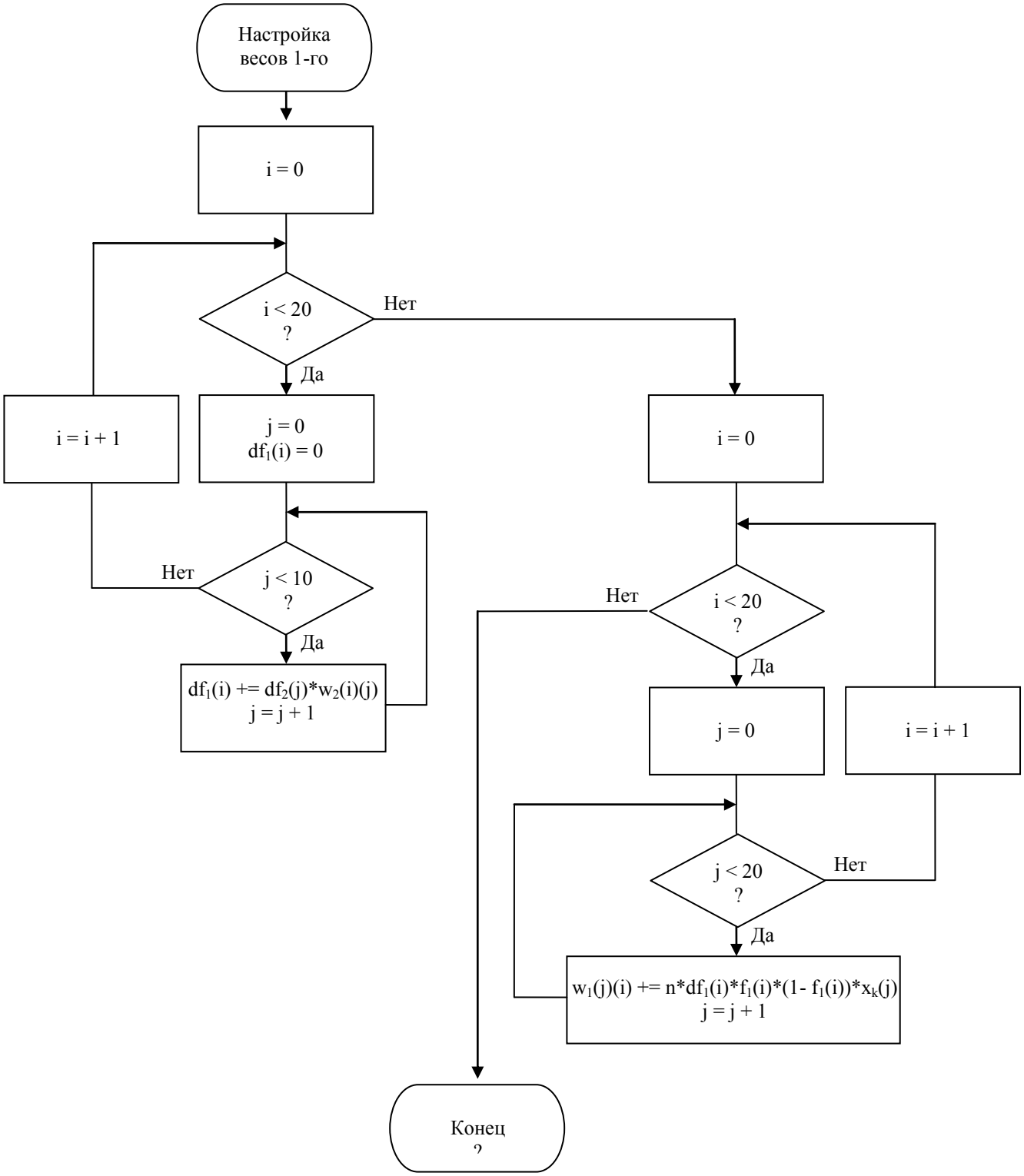

Ввод образа и целевых значений
Ввод образа:
$x_k=\{x_0,\ldots,x_{19}\}$
tt_k
i = 0
i < 10
?
Нет
Да
t(i) = 0
i = i + 1
$t(tt_k) = 1$
Конец

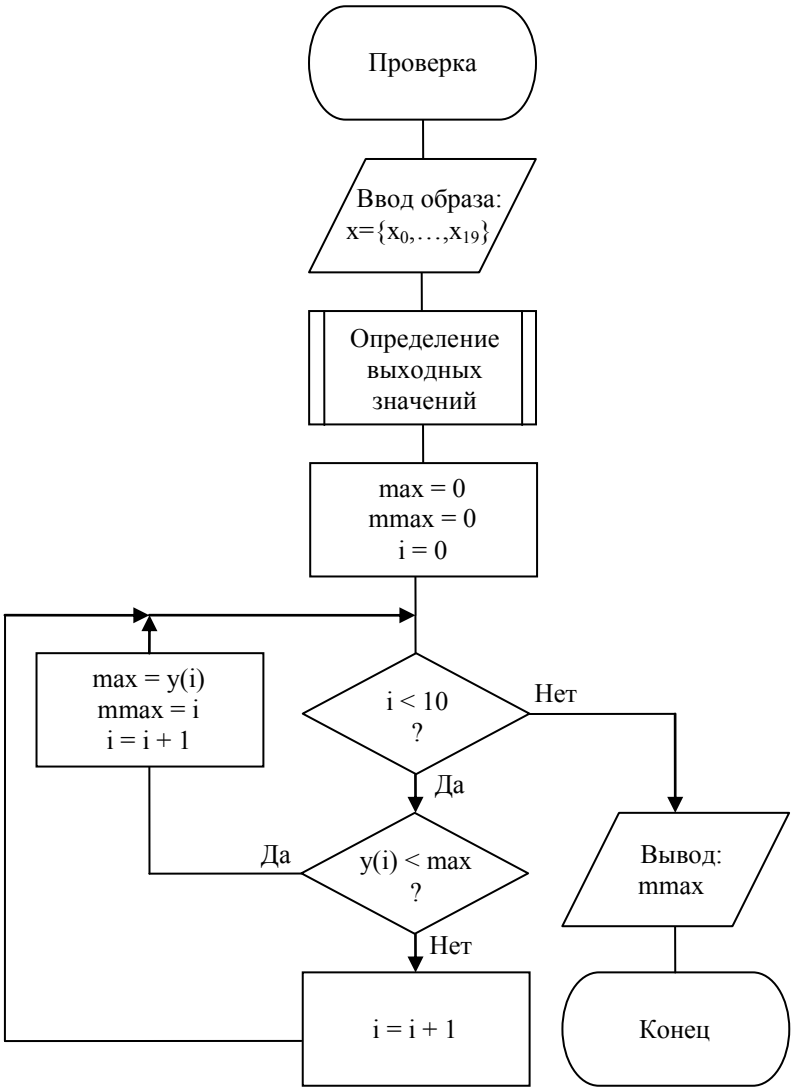

Определение выходных значений
i = 0
i < 20 ?
Нет
Да
j = 0
s = 0
f1(i) = 1/(1 + exp(-s))
i = i + 1
j < 20 ?
f1(i)
Да
s = s + xk(j)*w1(j)(i)
j = j + 1
i = 0
i < 10 ?
Нет
Да
j = 0
s = 0
y(i) = 1/(1 + exp(-s))
dy(i) = t(i) – y(i)
i = i + 1
j < 20 ?
Нет
Да
s = s + f2(j)*w3(j)(i)
j = j + 1
i = 0
i < 20 ?
Нет
Да
j = 0
s = 0
s = s + f1(j)*w2(j)(i)
j = j + 1
j < 20 ?
Нет
Да
s = s + f1(j)*w2(j)(i)
j = j + 1
Конец


Вычисление погрешности
s = 0
i = 0
i < 10 ?
Нет
Да
s = s + dy(i)²
i = i + 1
s = s / 2
|s| > e ?
Да
Нет
e = |s|
Конец


Настройка весовых коэффициентов
Настройка весов 3-го
Настройка весов 2-го
Настройка весов 1-го
Конец
Настройка весов 3-го
i = 0
i < 10 ?
Нет
Да
Конец
j = 0
i = i + 1
j < 20 ?
Нет
Да
w3(j)(i) += n*dy(i)*y(i)*(1-y(i))*f2(j)
j = j + 1

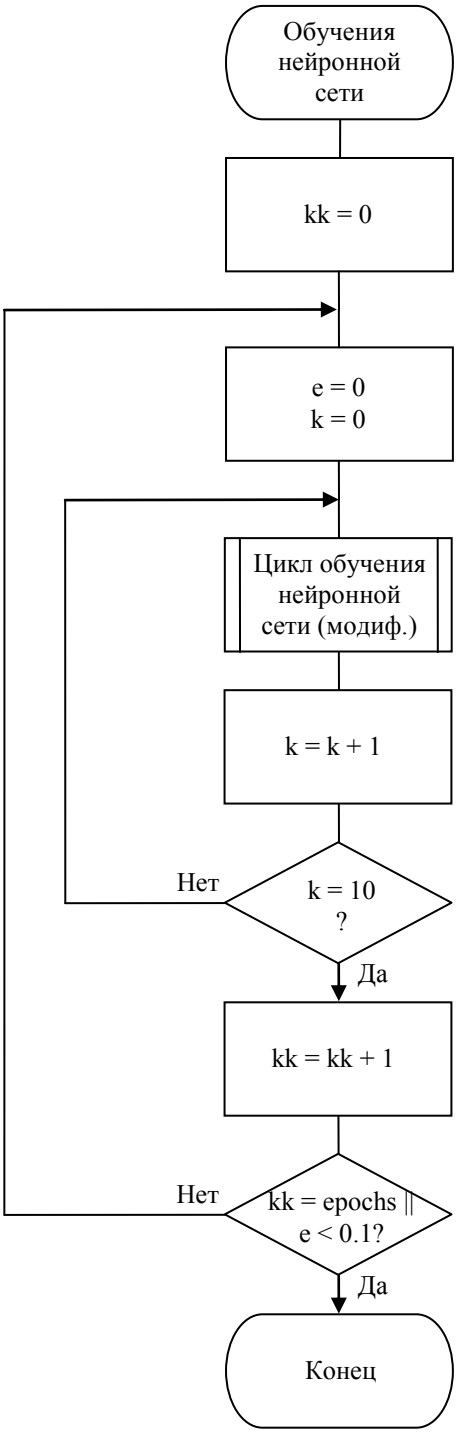

Настройка весов 2-го уровня
i = 0
i < 20 ?
Нет
Да
i = i + 1
j = 0
df2(i) = 0
j < 10 ?
Нет
Да
df2(i) += dy(j)*w3(i)(j)
j = j + 1
i = 0
i < 20 ?
Нет
Да
j = 0
i = i + 1
j < 20 ?
Нет
Да
w2(j)(i) += n*df2(i)*f2(i)*(1- f2(i))*f1(j)
j = j + 1
Конец

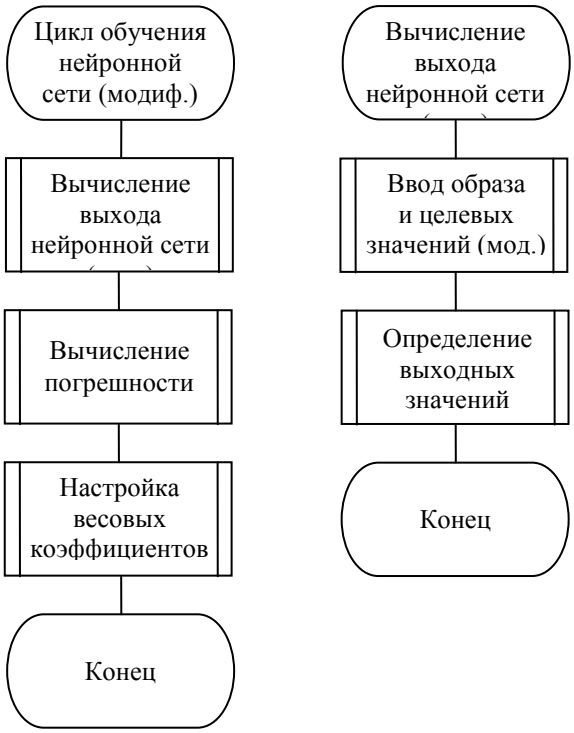

Настройка весов 1-го
i = 0
i < 20 ?
Нет
Да
i = i + 1
j = 0
df1(i) = 0
j < 10 ?
Нет
Да
df1(i) += df2(j)*w2(i)(j)
j = j + 1
i = 0
i < 20 ?
Нет
Да
j = 0
i = i + 1
j < 20 ?
Нет
Да
w1(j)(i) += n*df1(i)*f1(i)*(1- f1(i))*xk(j)
j = j + 1
Конец

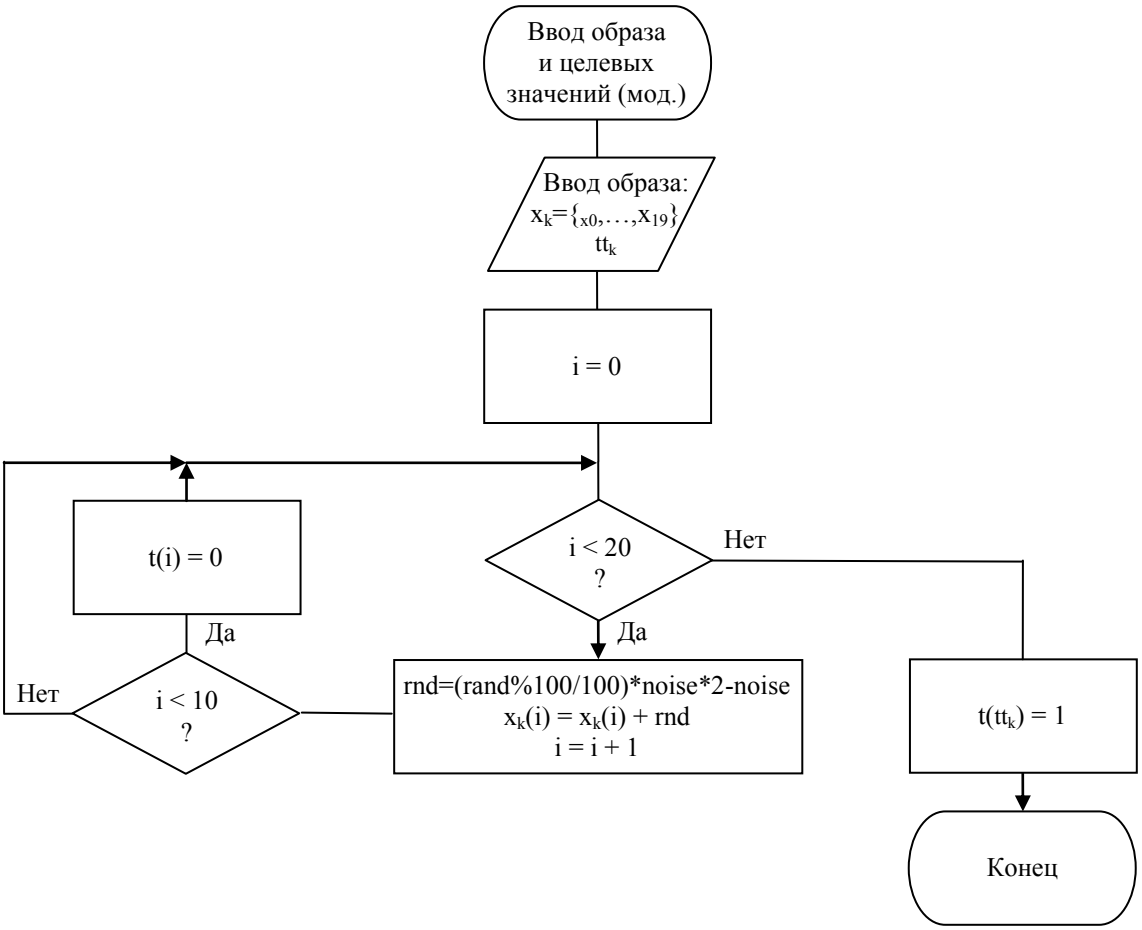

Проверка
Ввод образа:
x={x0,…,x19}
Определение выходных значений
max = 0
mmax = 0
i = 0
i < 10 ?
Нет
Да
max = y(i)
mmax = i
i = i + 1
Да
y(i) < max ?
Нет
Вывод:
mmax
i = i + 1
Конец

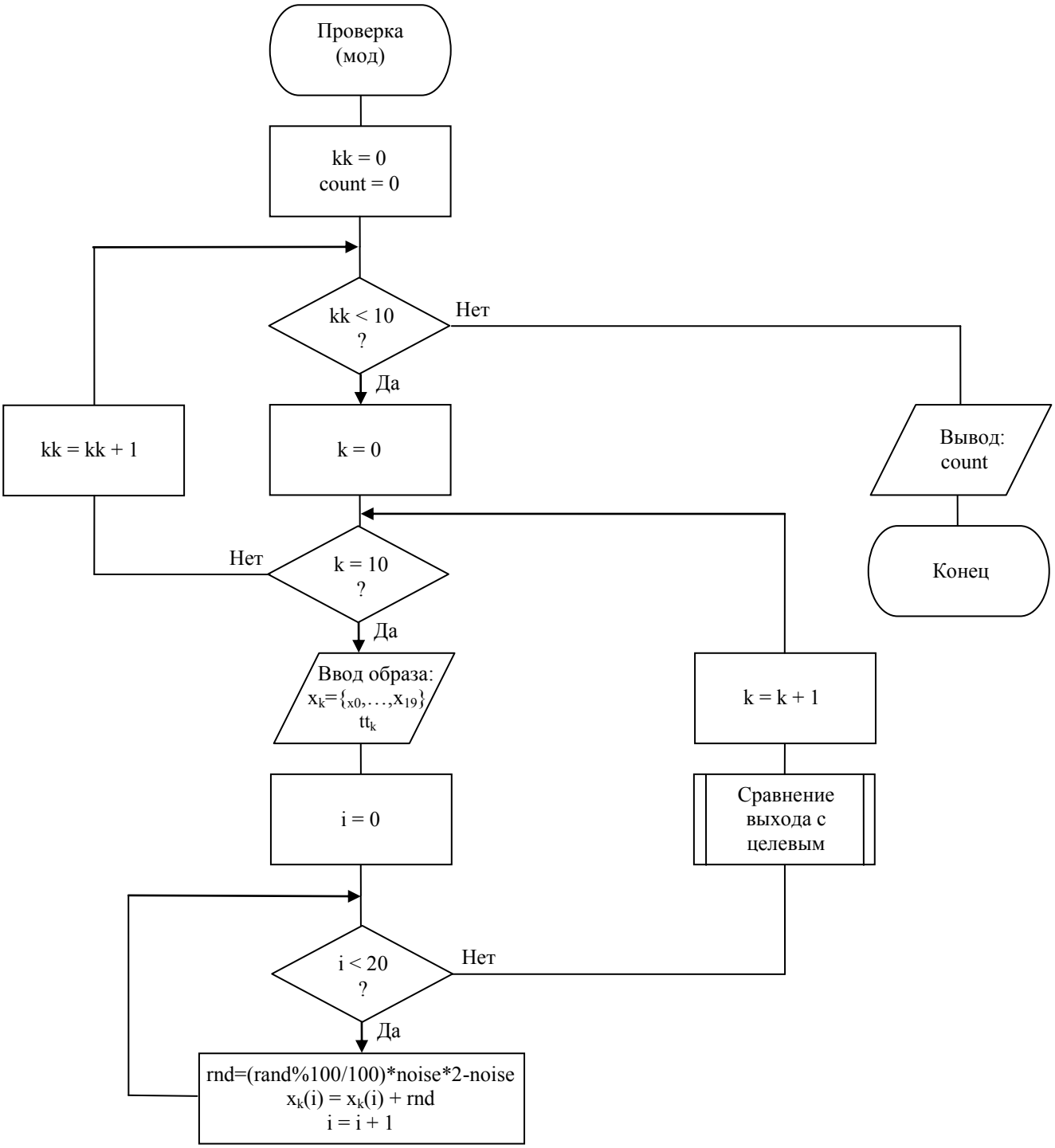
Приложение 3

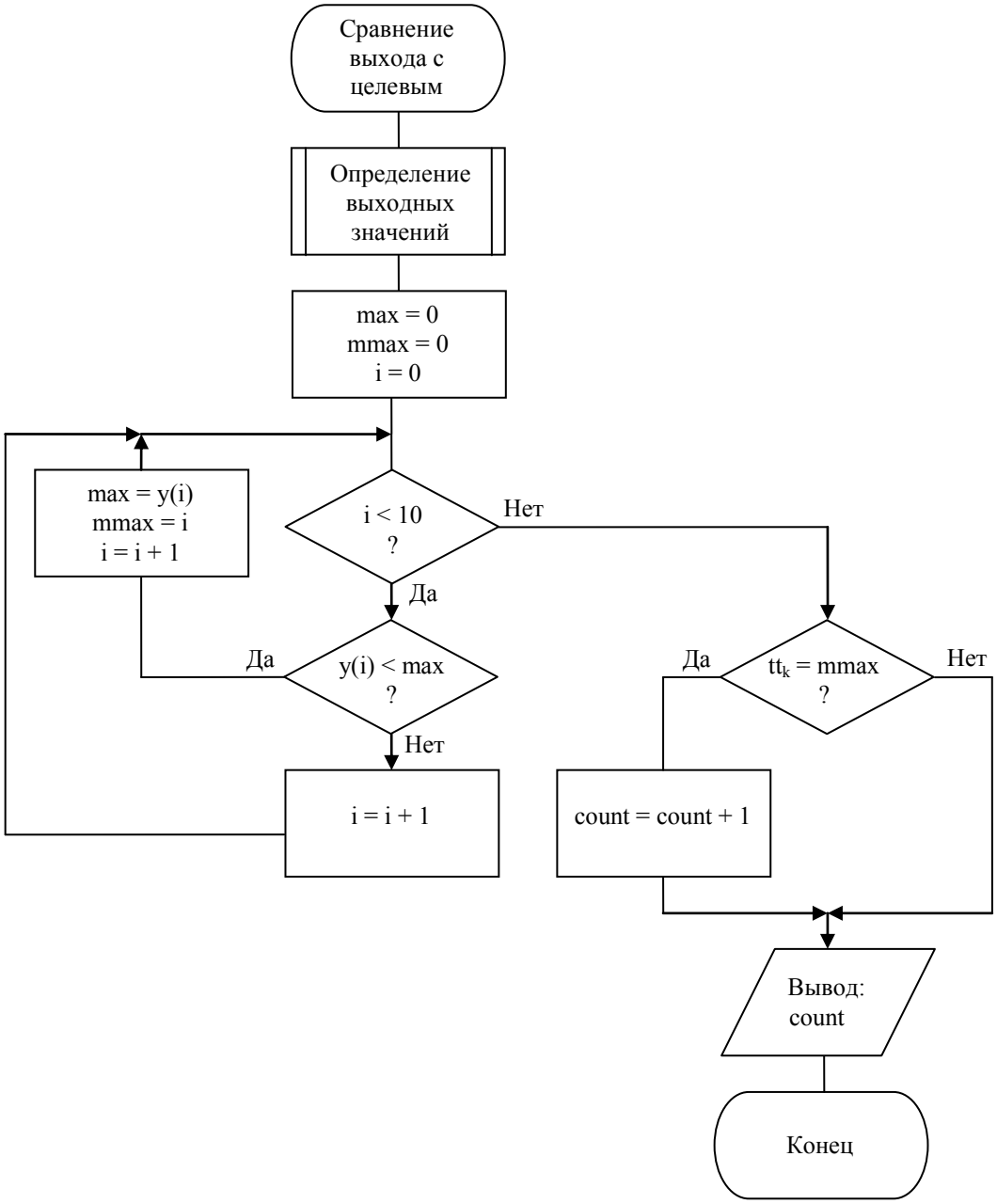
Блок-схема алгоритма для Лабораторной работы №3


Обучения нейронной сети
kk = 0
e = 0
k = 0
Цикл обучения нейронной сети (модиф.)
k = k + 1
Нет
k = 10 ?
Да
kk = kk + 1
Нет
kk = epochs || e < 0.1?
Да
Конец


Цикл обучения нейронной сети (модиф.)
Вычисление выхода нейронной сети
Вычисление погрешности
Настройка весовых коэффициентов
Конец
Вычисление выхода нейронной сети
Ввод образа и целевых значений (мод.)
Определение выходных значений
Конец

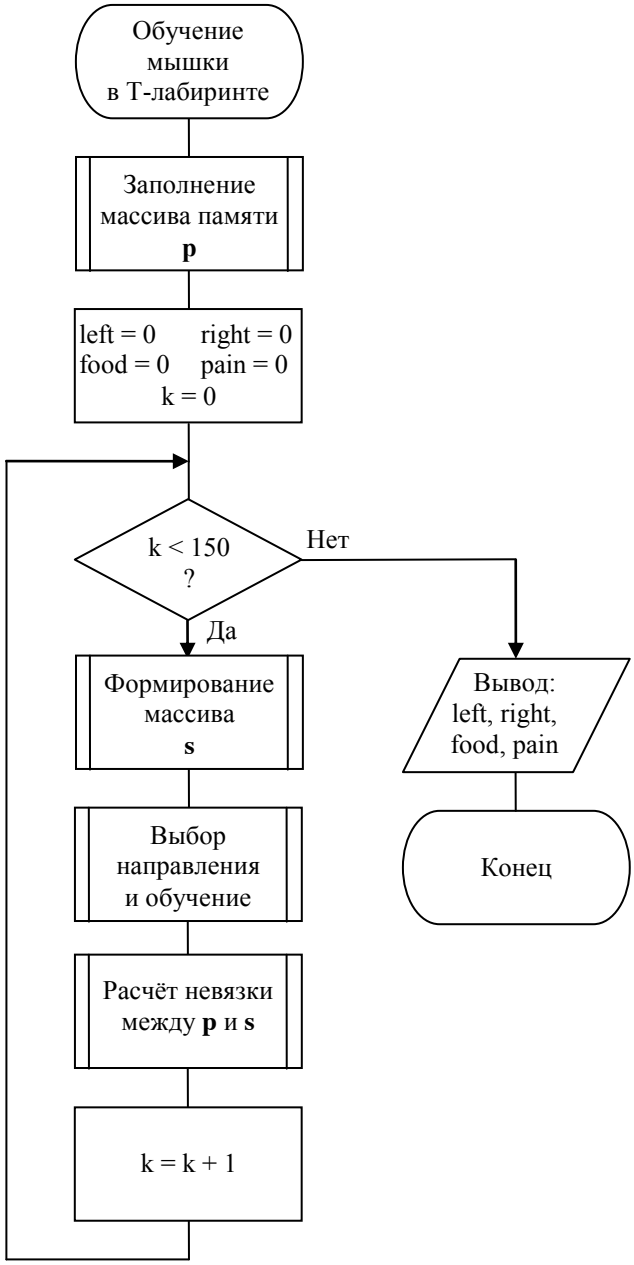

Ввод образа и целевых значений (мод.)
Ввод образа:
$x_k=\{x_0,\ldots,x_{19}\}$
tt_k
i = 0
i < 20 ?
Нет
Да
t(i) = 0
Да
Нет
i < 10 ?
rnd=(rand%100/100)*noise*2-noise
$x_k(i) = x_k(i) + rnd$
i = i + 1
$t(tt_k) = 1$
Конец

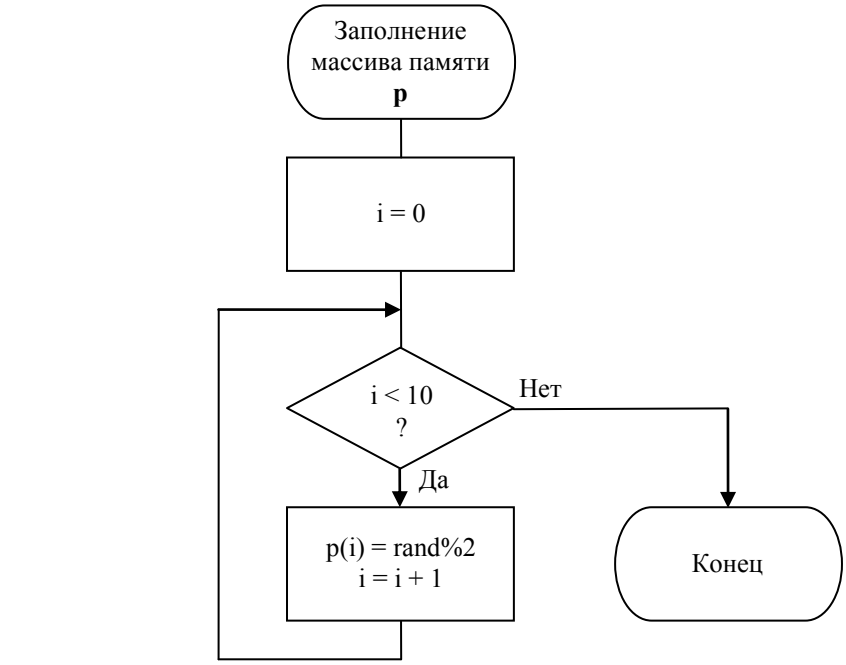

Проверка (мод)
kk = 0
count = 0
kk < 10 ?
Нет
Да
kk = kk + 1
k = 0
Вывод: count
Конец
Нет
k = 10 ?
Да
Ввод образа:
$x_k=\{x_0,\ldots,x_{19}\}$
tt_k
k = k + 1
i = 0
Сравнение выхода с целевым
i < 20 ?
Нет
Да
rnd=(rand%100/100)*noise*2-noise
$x_k(i) = x_k(i) + rnd$
i = i + 1

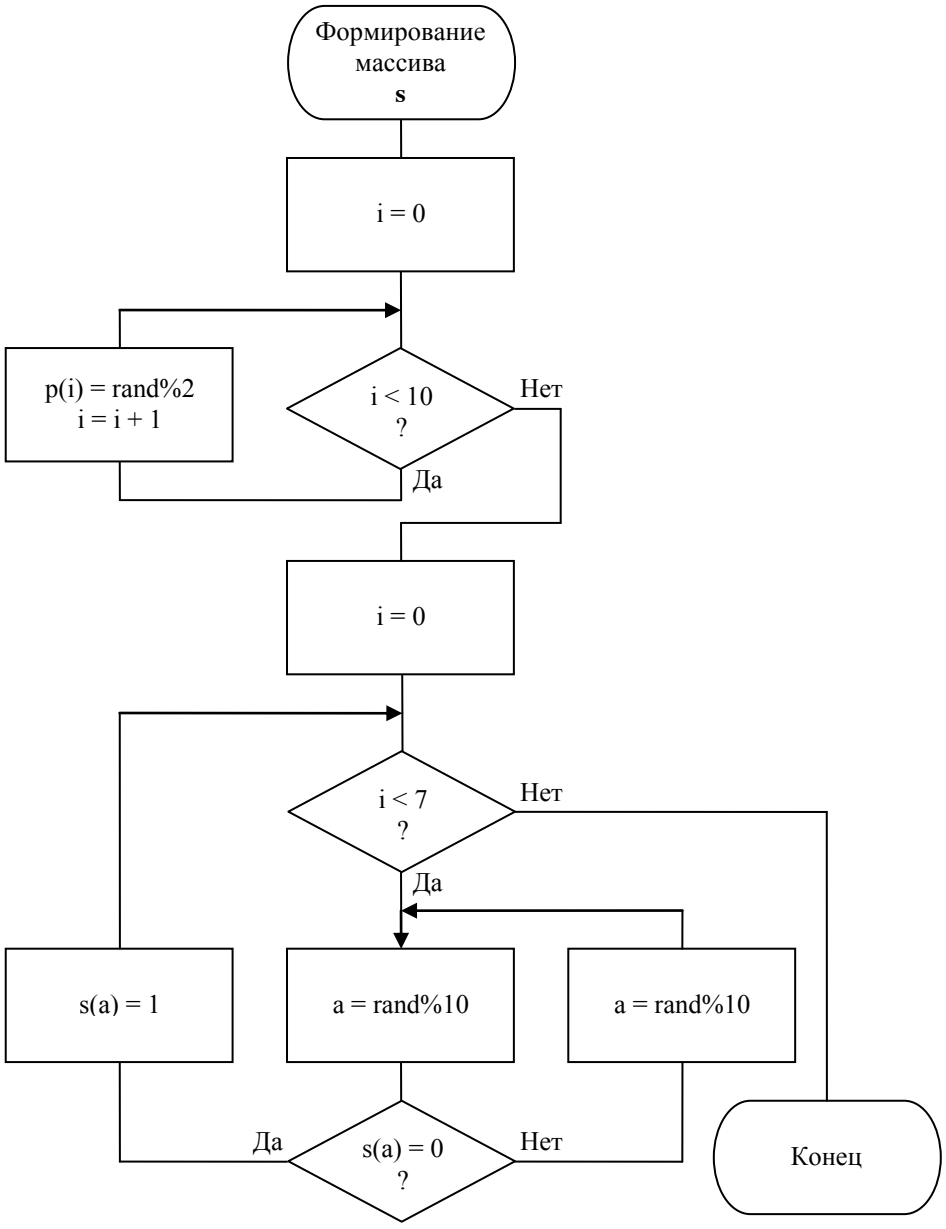

Сравнение выхода с целевым
Определение выходных значений
max = 0
mmax = 0
i = 0
i < 10 ?
Нет
Да
max = y(i)
mmax = i
i = i + 1
Да
y(i) < max ?
Нет
i = i + 1
Да
tt_k = mmax ?
Нет
count = count + 1
Вывод: count
Конец

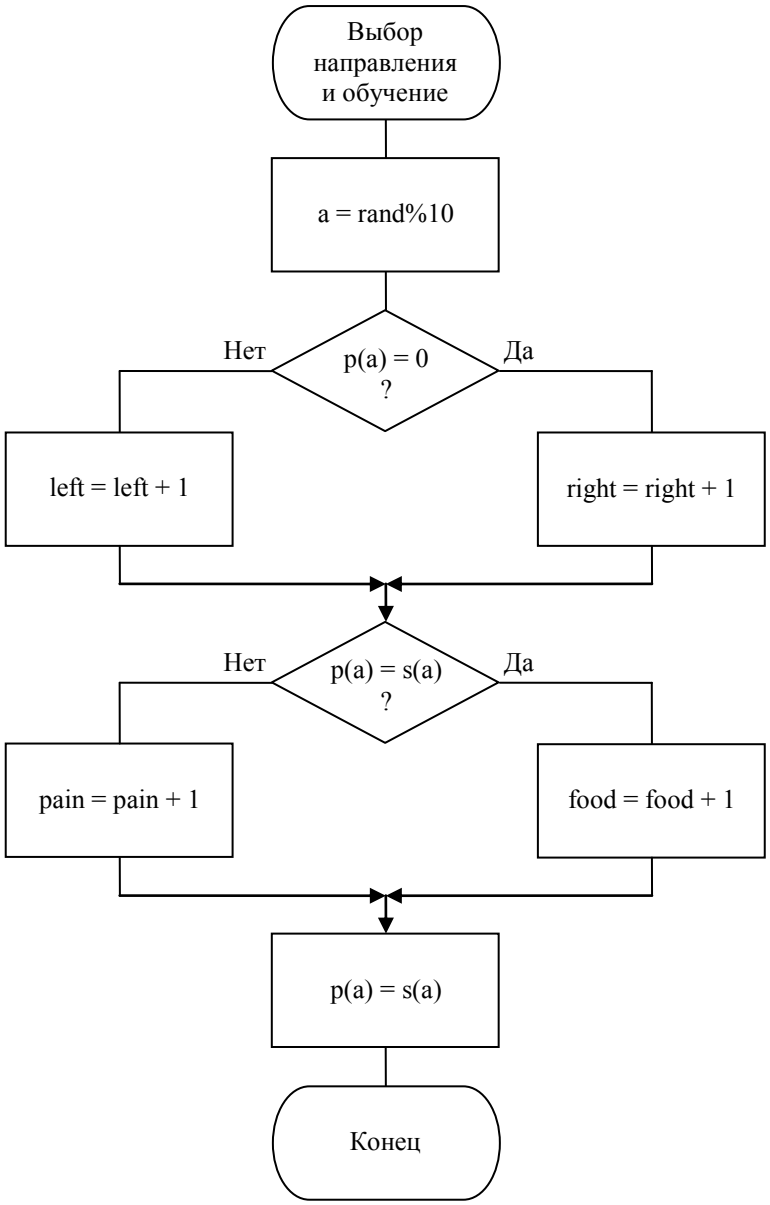
Приложение 4

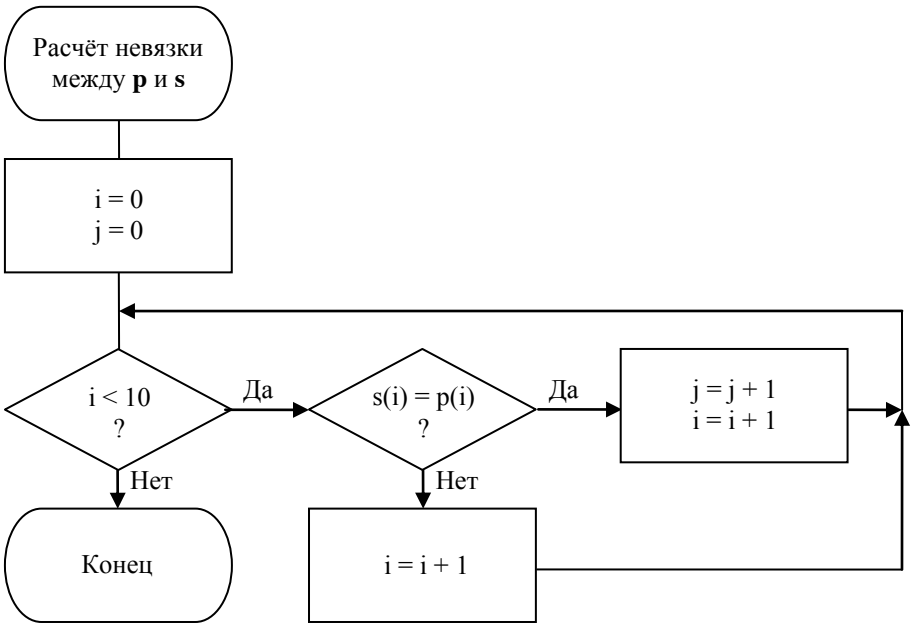
Блок-схема алгоритма для Лабораторной работы №4




Заполнение
массива памяти
p
i = 0
i < 10
?
Нет
Да
p(i) = rand%2
i = i + 1
Конец

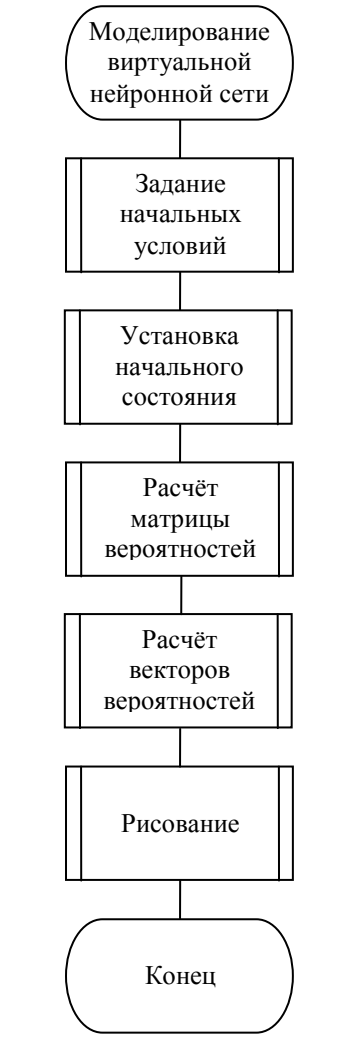

Формирование
массива
s
i = 0
p(i) = rand%2
i = i + 1
i < 10
?
Нет
Да
i = 0
i < 7
?
Нет
Да
s(a) = 1
a = rand%10
a = rand%10
Да
s(a) = 0
?
Нет
Конец

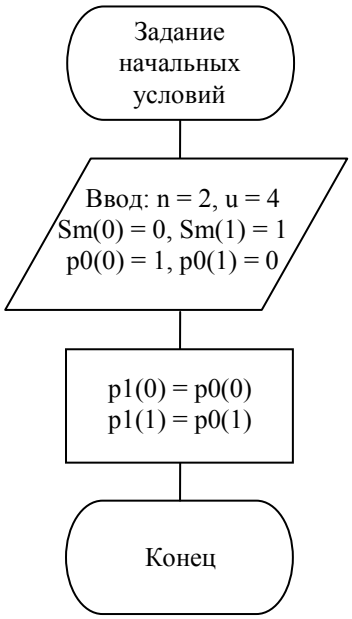

Выбор направления и обучение
a = rand%10
Нет
p(a) = 0 ?
Да
left = left + 1
right = right + 1
Нет
p(a) = s(a) ?
Да
pain = pain + 1
food = food + 1
p(a) = s(a)
Конец

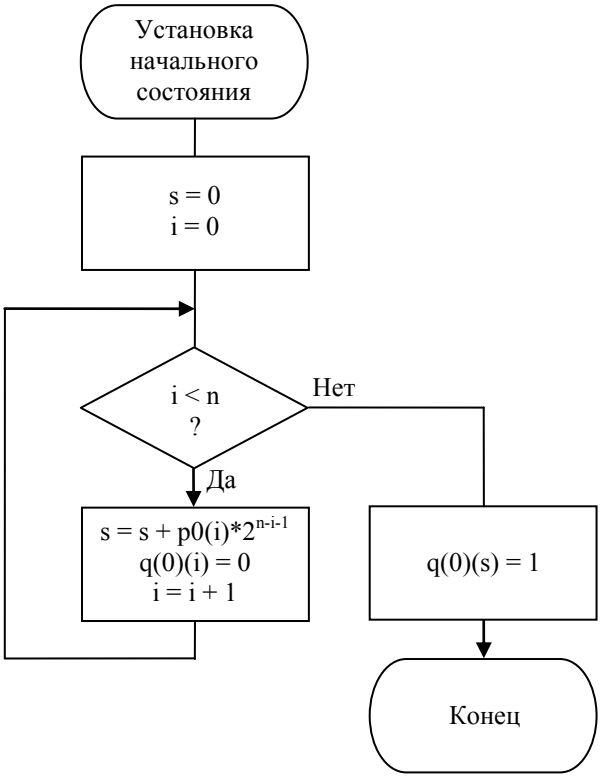

Расчёт невязки между **p** и **s**
i = 0
j = 0
i < 10 ?
Да
s(i) = p(i) ?
Да
j = j + 1
i = i + 1
Нет
Нет
Конец
i = i + 1

Приложение 5

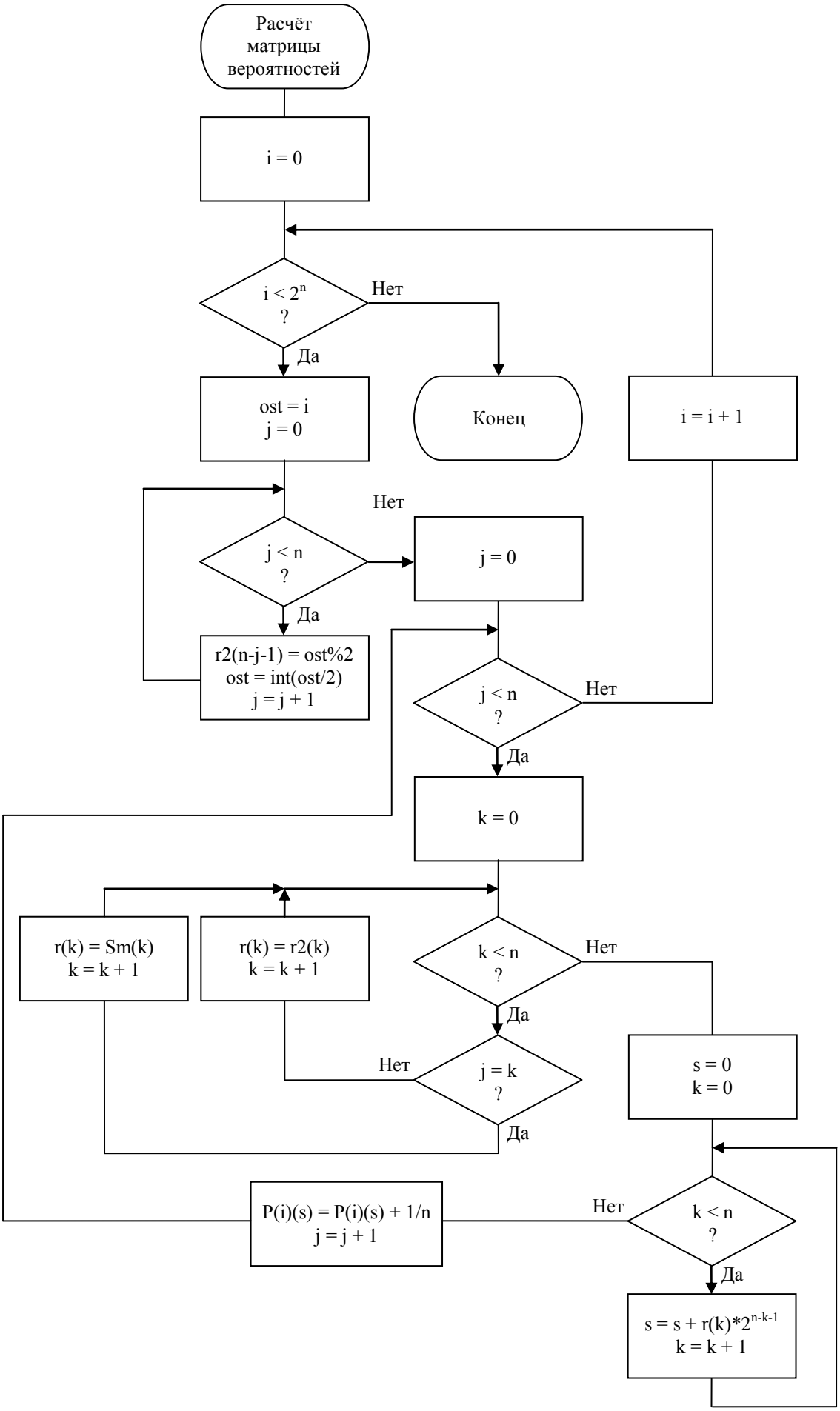
Блок-схема алгоритма для Лабораторной работы №5

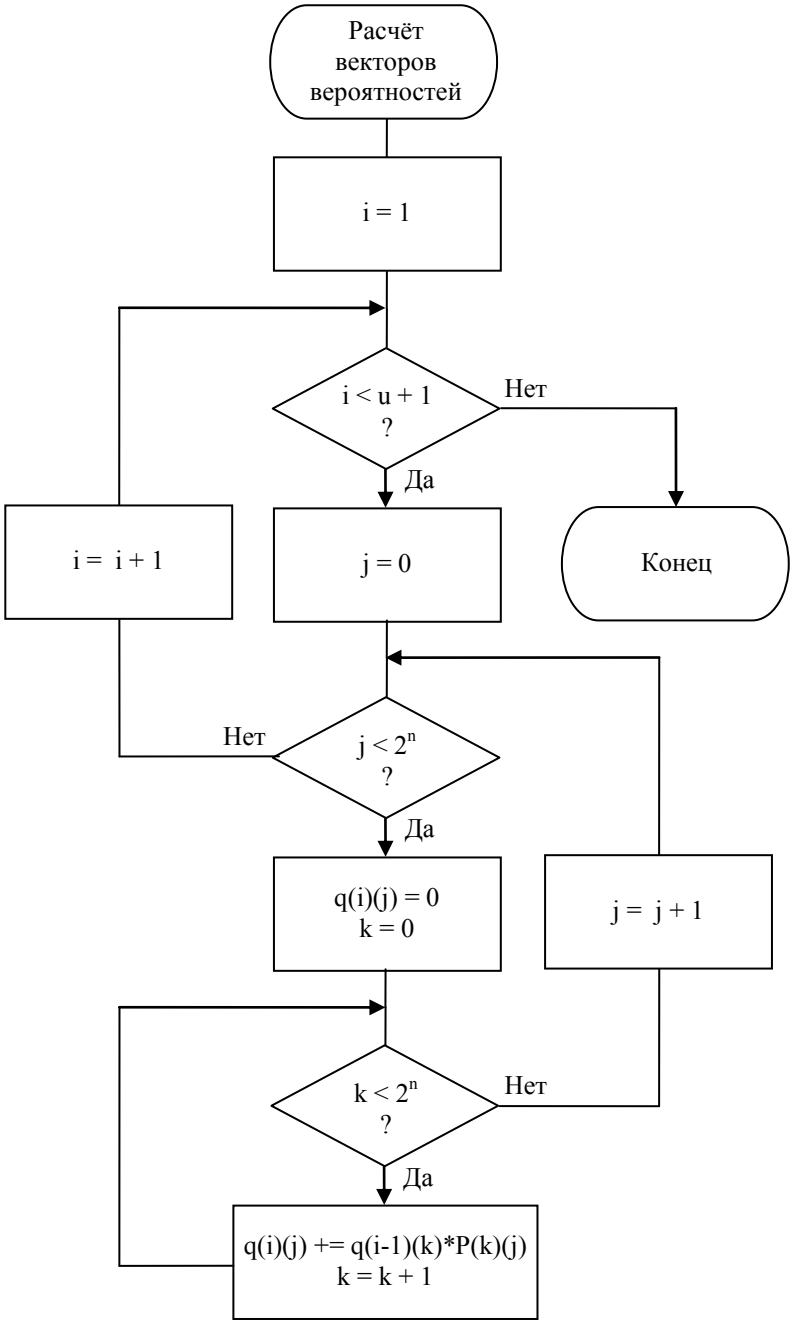




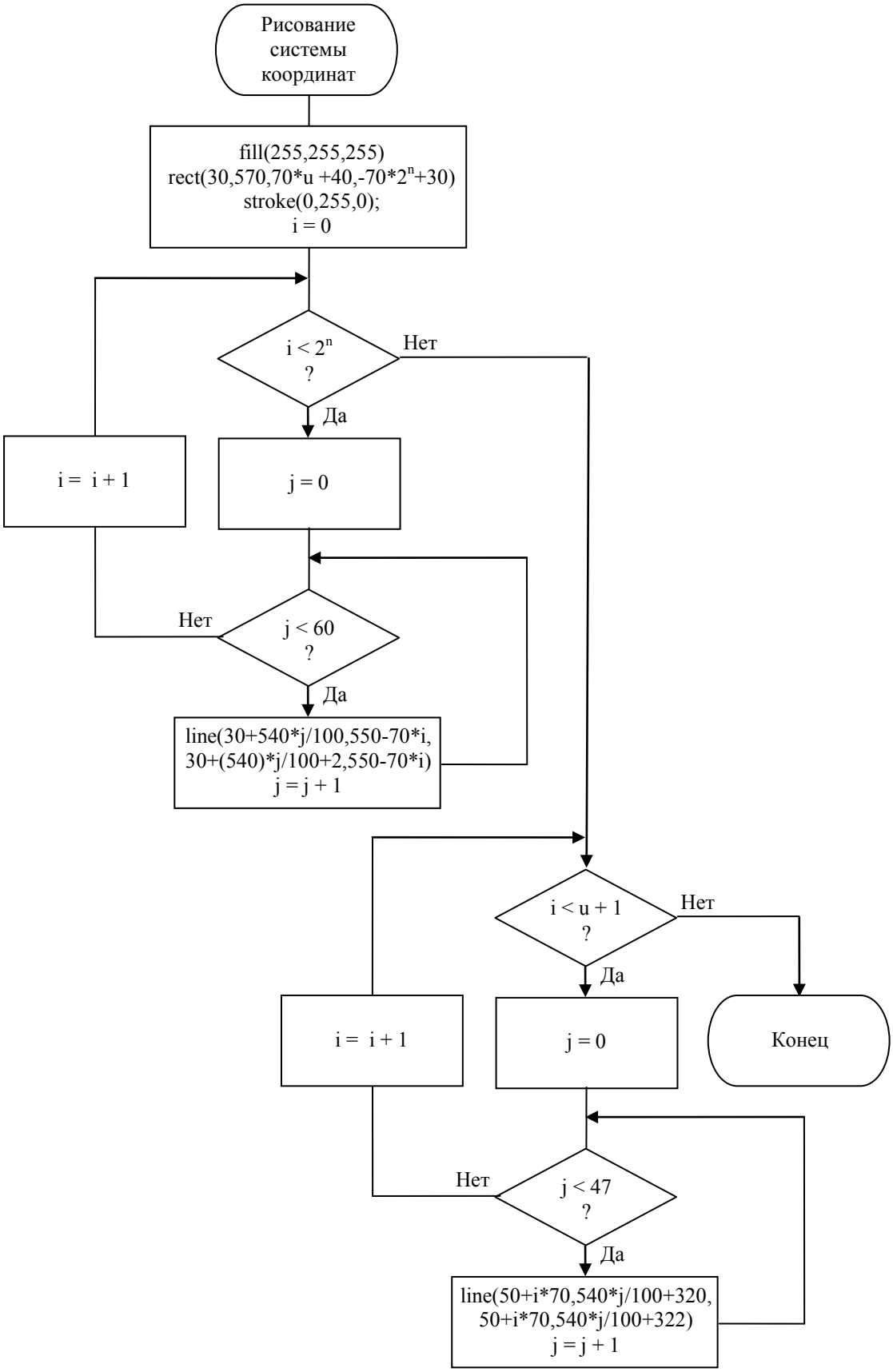

Установка начального состояния
s = 0
i = 0
i < n ?
Нет
Да
s = s + p0(i)*2^n-i-1
q(0)(i) = 0
i = i + 1
q(0)(s) = 1
Конец

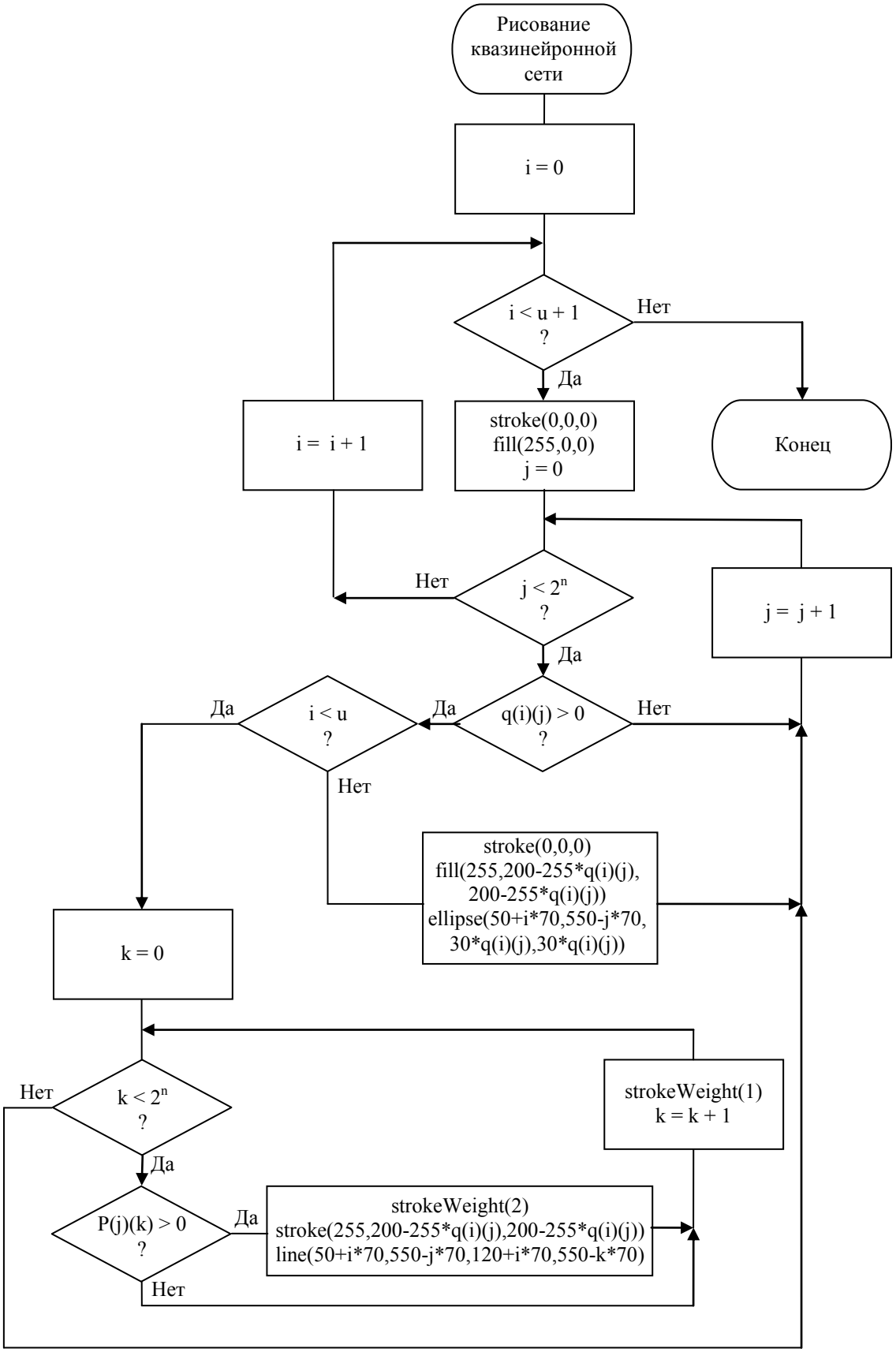

Рисование
Рисование системы координат
Рисование квазинейронной сети
Конец


Расчёт матрицы вероятностей
i = 0
i < 2^n ?
Нет
Да
Конец
ost = i
j = 0
i = i + 1
Нет
j < n ?
j = 0
Да
r2(n-j-1) = ost%2
ost = int(ost/2)
j = j + 1
j < n ?
Нет
Да
k = 0
r(k) = Sm(k)
k = k + 1
r(k) = r2(k)
k = k + 1
k < n ?
Нет
Да
Нет
j = k ?
Да
s = 0
k = 0
P(i)(s) = P(i)(s) + 1/n
j = j + 1
Нет
k < n ?
Да
s = s + r(k)*2^n-k-1
k = k + 1



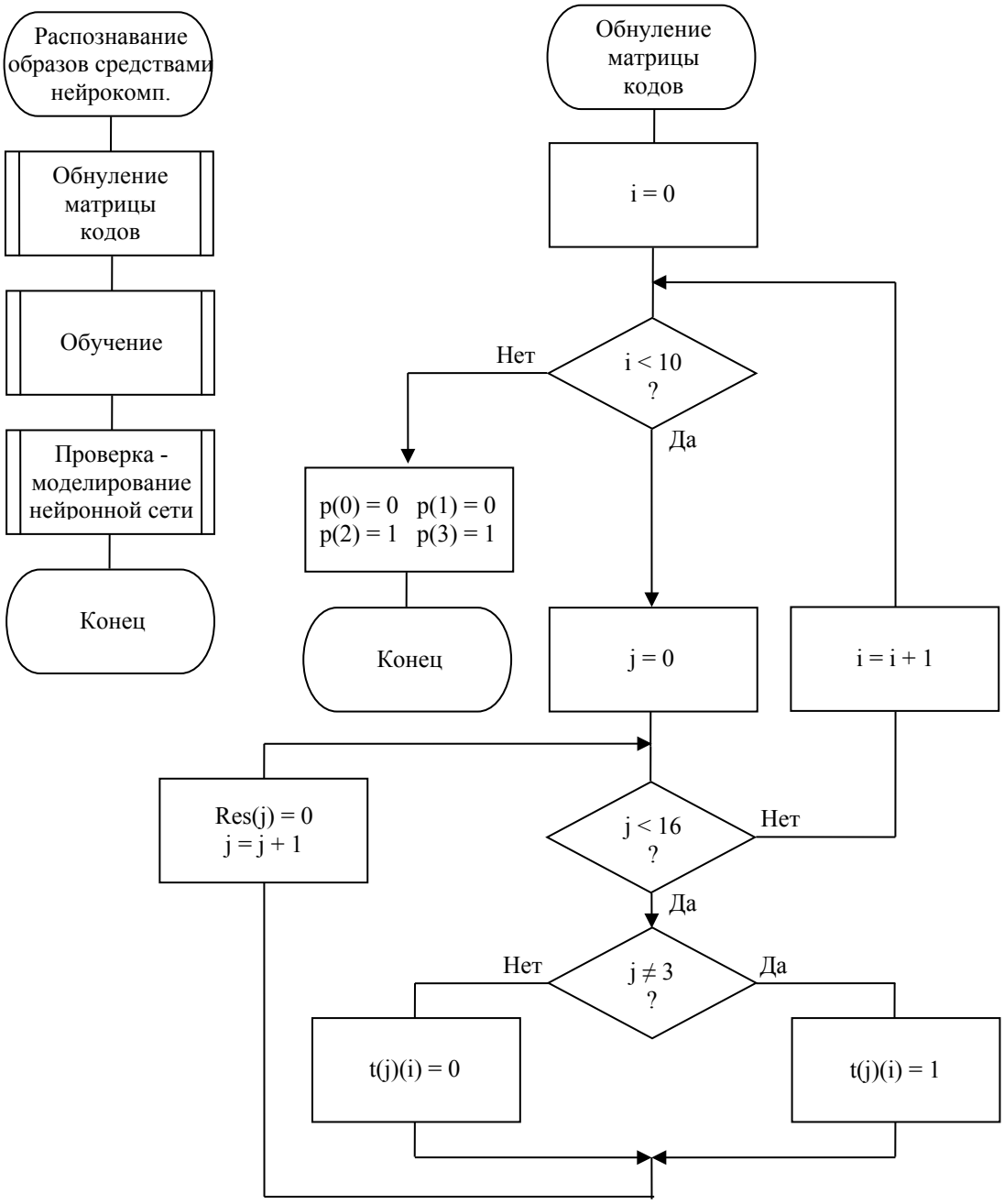
Обозначения:
n – разрядность машины (количество столбцов сенсорной матрицы)
u – системное квантованное время
Sm(n) – сенсорная матрица
p0(n) – внутренняя память нейрокомпьютера
p1(n) – внутренняя память нейрокомпьютера (для сохранения состояния)
$P(2^n)(2^n)$ – матрица переходов
$q(u)(2^n)$ – вектора вероятностей
ost – остаток от деления
rect(x1,y1,x2-x1,y2-y1) – прямоугольник
line(x1,y1,x2,y2) – линия
ellipse(x,y,r1,r2) - эллипс
fill(R,G,B) – цвет заливки
stroke(R,G,B) – цвет линии
strokeWeight(x) – толщина линии

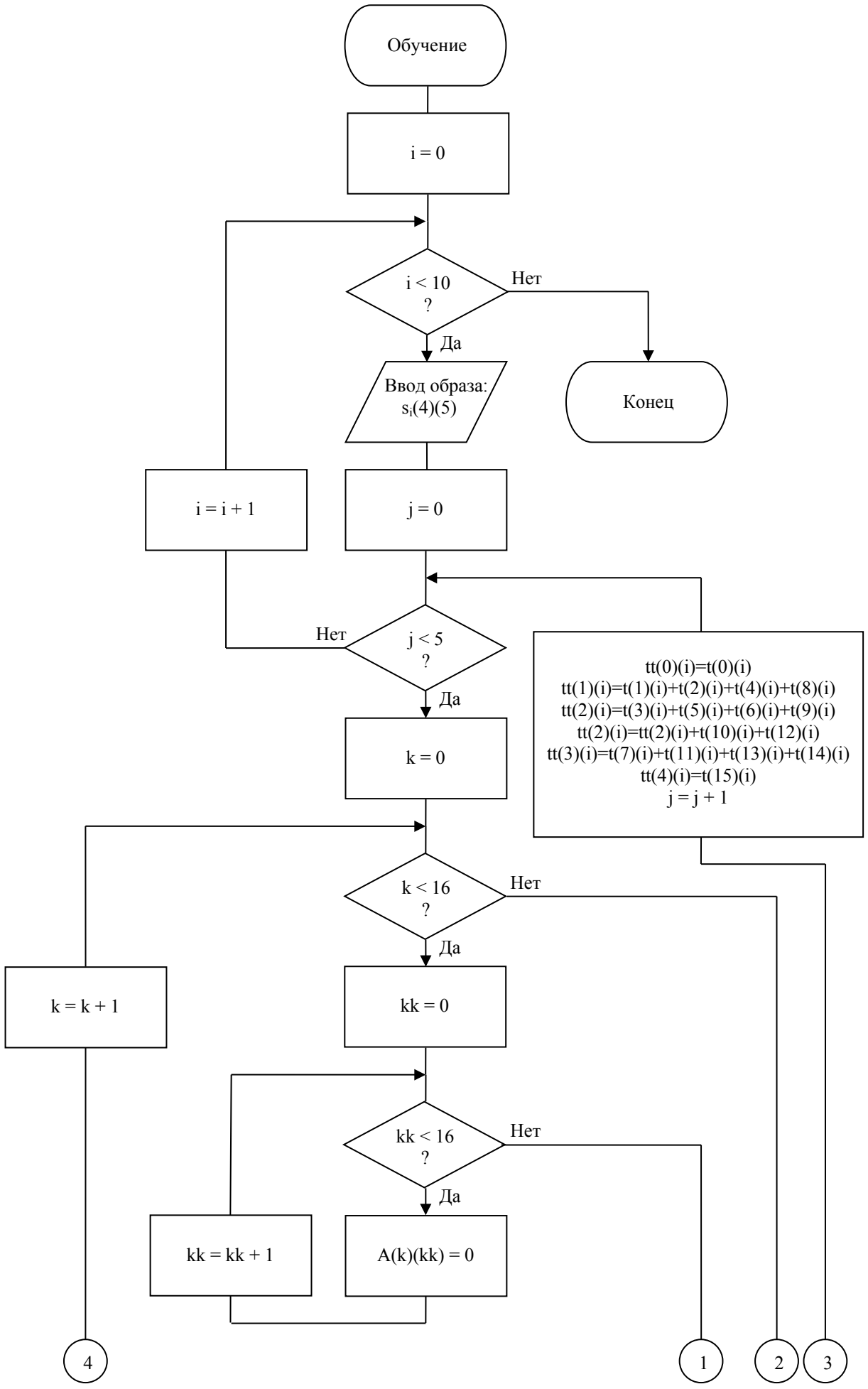

Рисование системы координат
fill(255,255,255)
rect(30,570,70*u +40,-70*2^n+30)
stroke(0,255,0);
i = 0
i < 2^n ?
Нет
Да
i = i + 1
j = 0
Нет
j < 60 ?
Да
line(30+540*j/100,550-70*i,
30+(540)*j/100+2,550-70*i)
j = j + 1
i < u + 1 ?
Нет
Да
i = i + 1
j = 0
Конец
Нет
j < 47 ?
Да
line(50+i*70,540*j/100+320,
50+i*70,540*j/100+322)
j = j + 1

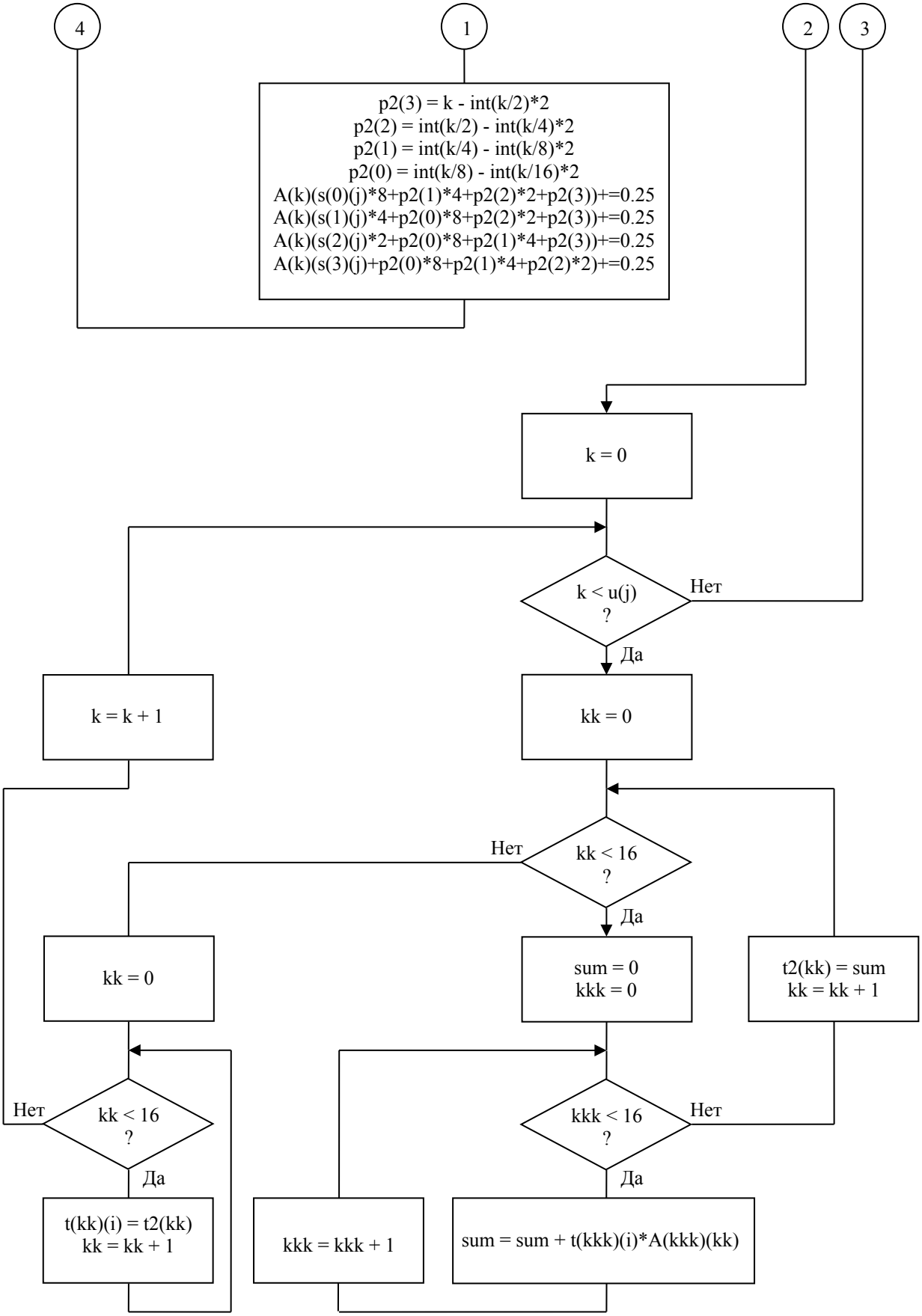

Рисование квазинейронной сети
i = 0
i < u + 1 ?
Нет
Да
Конец
stroke(0,0,0)
fill(255,0,0)
j = 0
i = i + 1
j < 2^n ?
Нет
Да
j = j + 1
q(i)(j) > 0 ?
Да
Нет
i < u ?
Да
Нет
stroke(0,0,0)
fill(255,200-255*q(i)(j),
200-255*q(i)(j))
ellipse(50+i*70,550-j*70,
30*q(i)(j),30*q(i)(j))
k = 0
k < 2^n ?
Нет
Да
strokeWeight(1)
k = k + 1
P(j)(k) > 0 ?
Да
strokeWeight(2)
stroke(255,200-255*q(i)(j),200-255*q(i)(j))
line(50+i*70,550-j*70,120+i*70,550-k*70)
Нет

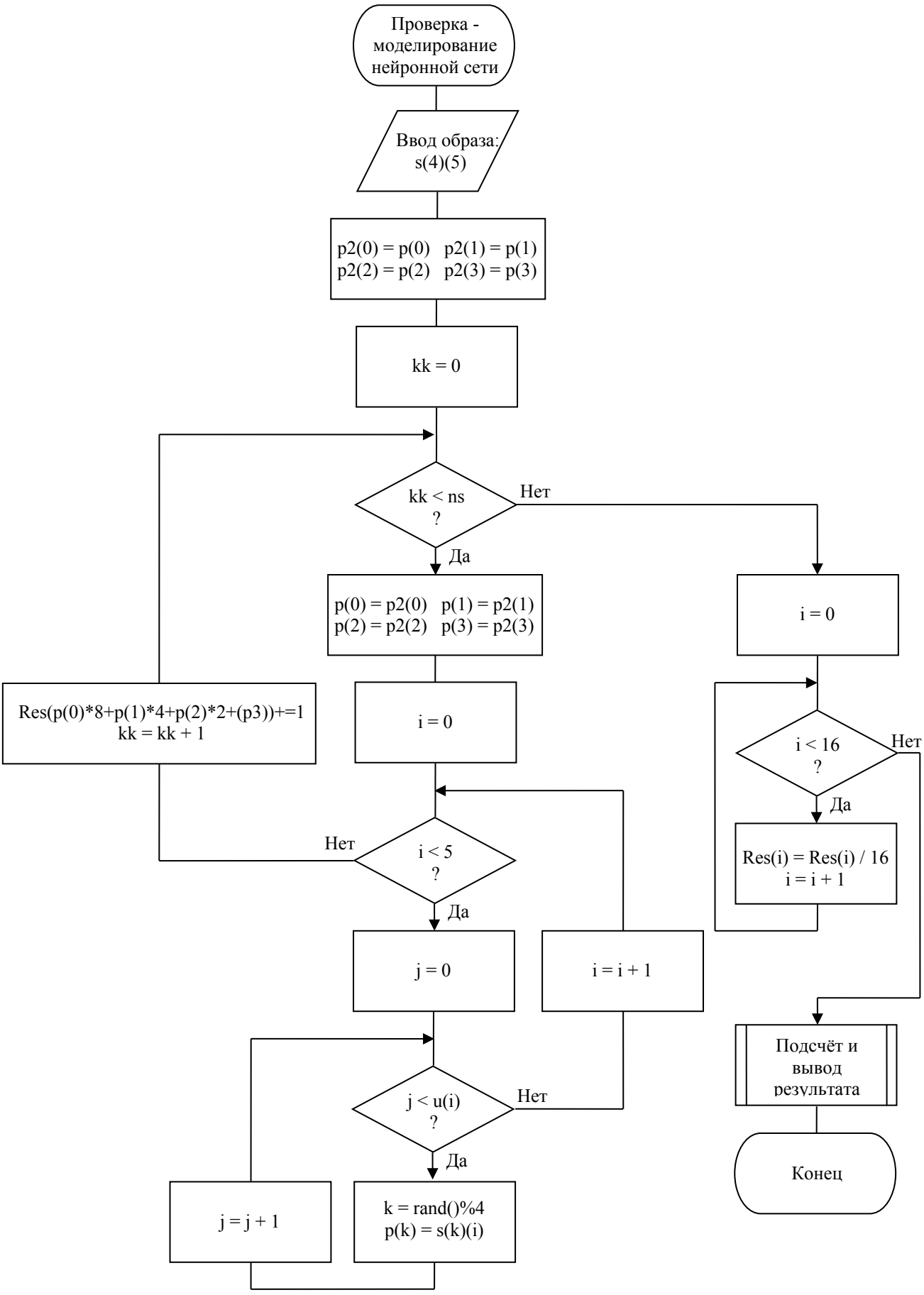
Приложение 6

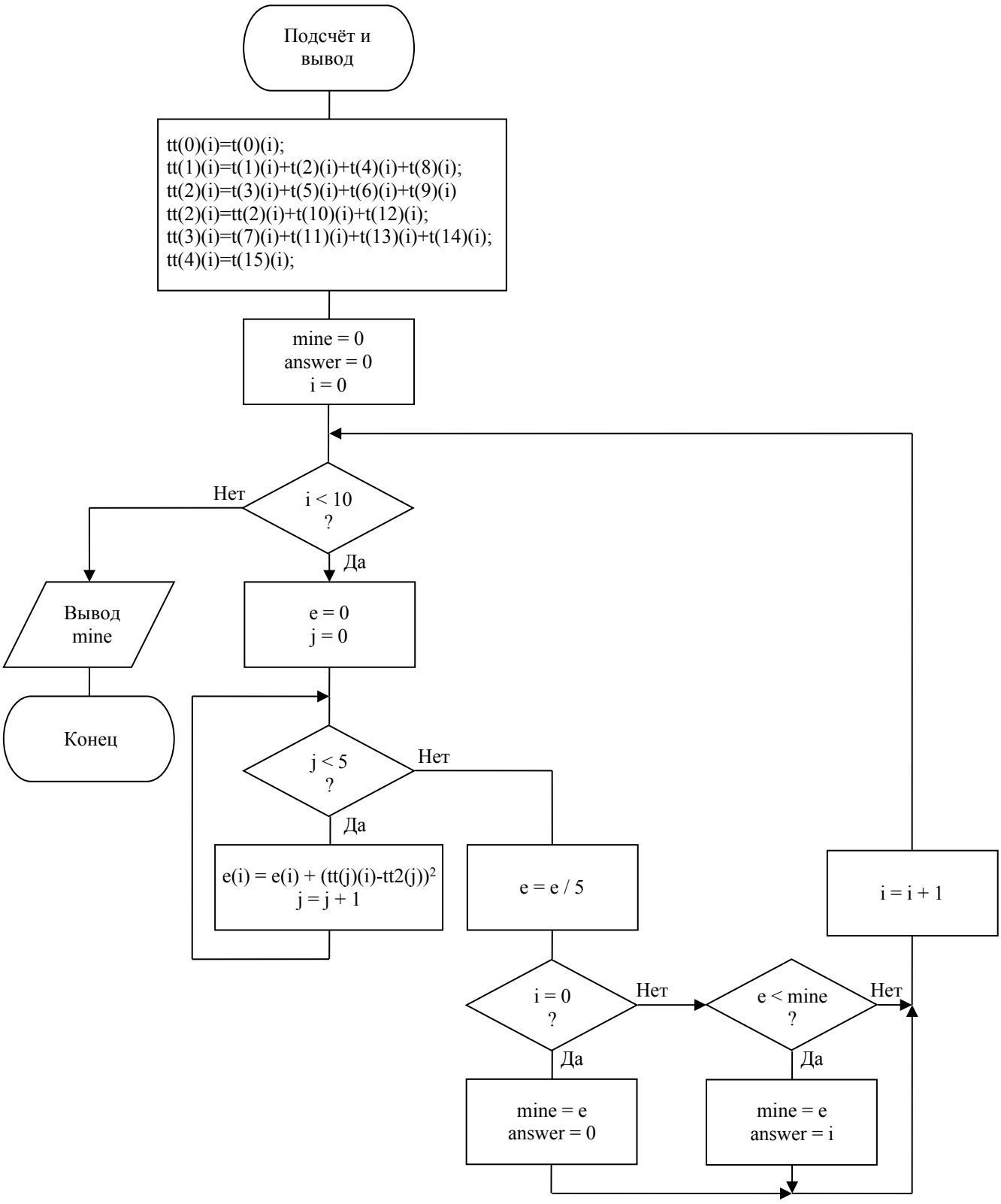
Блок-схема алгоритма для Лабораторной работы №6



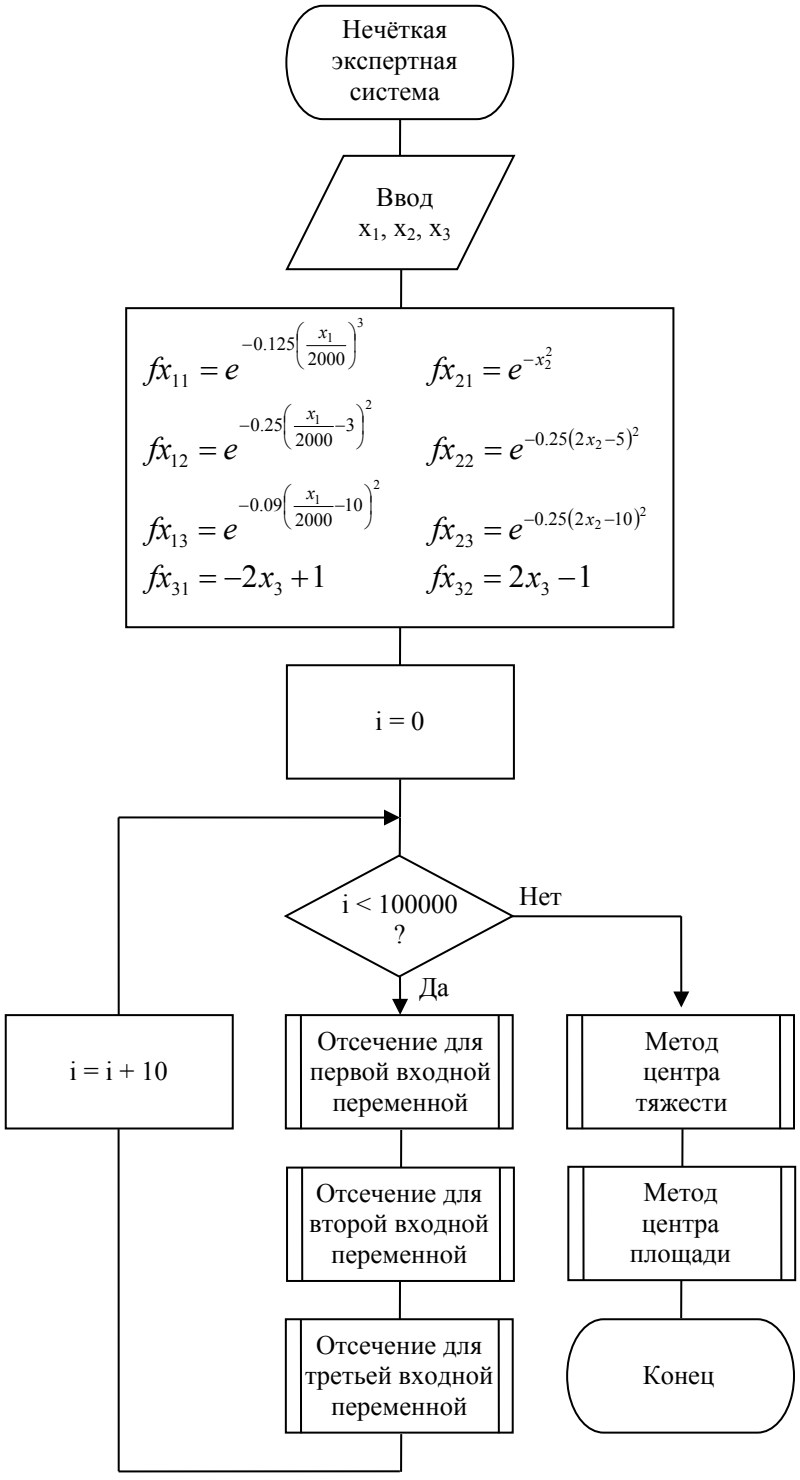

Обучение
i = 0
i < 10 ?
Нет
Да
Ввод образа: $s_i(4)(5)$
Конец
i = i + 1
j = 0
Нет
j < 5 ?
Да
tt(0)(i)=t(0)(i)
tt(1)(i)=t(1)(i)+t(2)(i)+t(4)(i)+t(8)(i)
tt(2)(i)=t(3)(i)+t(5)(i)+t(6)(i)+t(9)(i)
tt(2)(i)=tt(2)(i)+t(10)(i)+t(12)(i)
tt(3)(i)=t(7)(i)+t(11)(i)+t(13)(i)+t(14)(i)
tt(4)(i)=t(15)(i)
j = j + 1
k = 0
k < 16 ?
Нет
Да
k = k + 1
kk = 0
kk < 16 ?
Нет
Да
kk = kk + 1
A(k)(kk) = 0
4
1
2
3

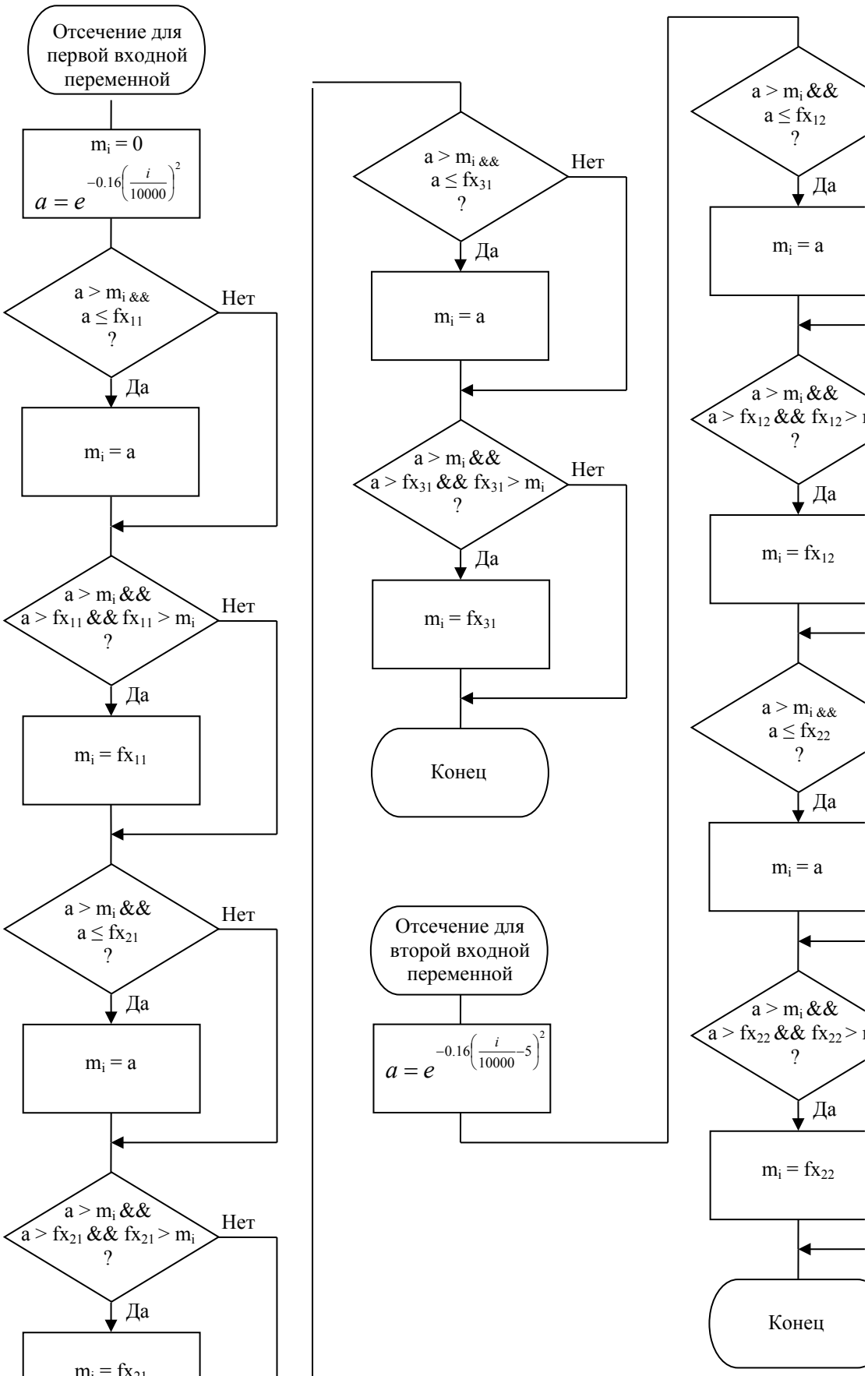

4
1
2
3
p2(3) = k - int(k/2)*2
p2(2) = int(k/2) - int(k/4)*2
p2(1) = int(k/4) - int(k/8)*2
p2(0) = int(k/8) - int(k/16)*2
A(k)(s(0)(j)*8+p2(1)*4+p2(2)*2+p2(3))+=0.25
A(k)(s(1)(j)*4+p2(0)*8+p2(2)*2+p2(3))+=0.25
A(k)(s(2)(j)*2+p2(0)*8+p2(1)*4+p2(3))+=0.25
A(k)(s(3)(j)+p2(0)*8+p2(1)*4+p2(2)*2)+=0.25
k = 0
k < u(j) ?
Нет
Да
k = k + 1
kk = 0
kk < 16 ?
Нет
Да
kk = 0
sum = 0
kkk = 0
t2(kk) = sum
kk = kk + 1
kk < 16 ?
Нет
Да
kkk < 16 ?
Нет
Да
t(kk)(i) = t2(kk)
kk = kk + 1
kkk = kkk + 1
sum = sum + t(kkk)(i)*A(kkk)(kk)

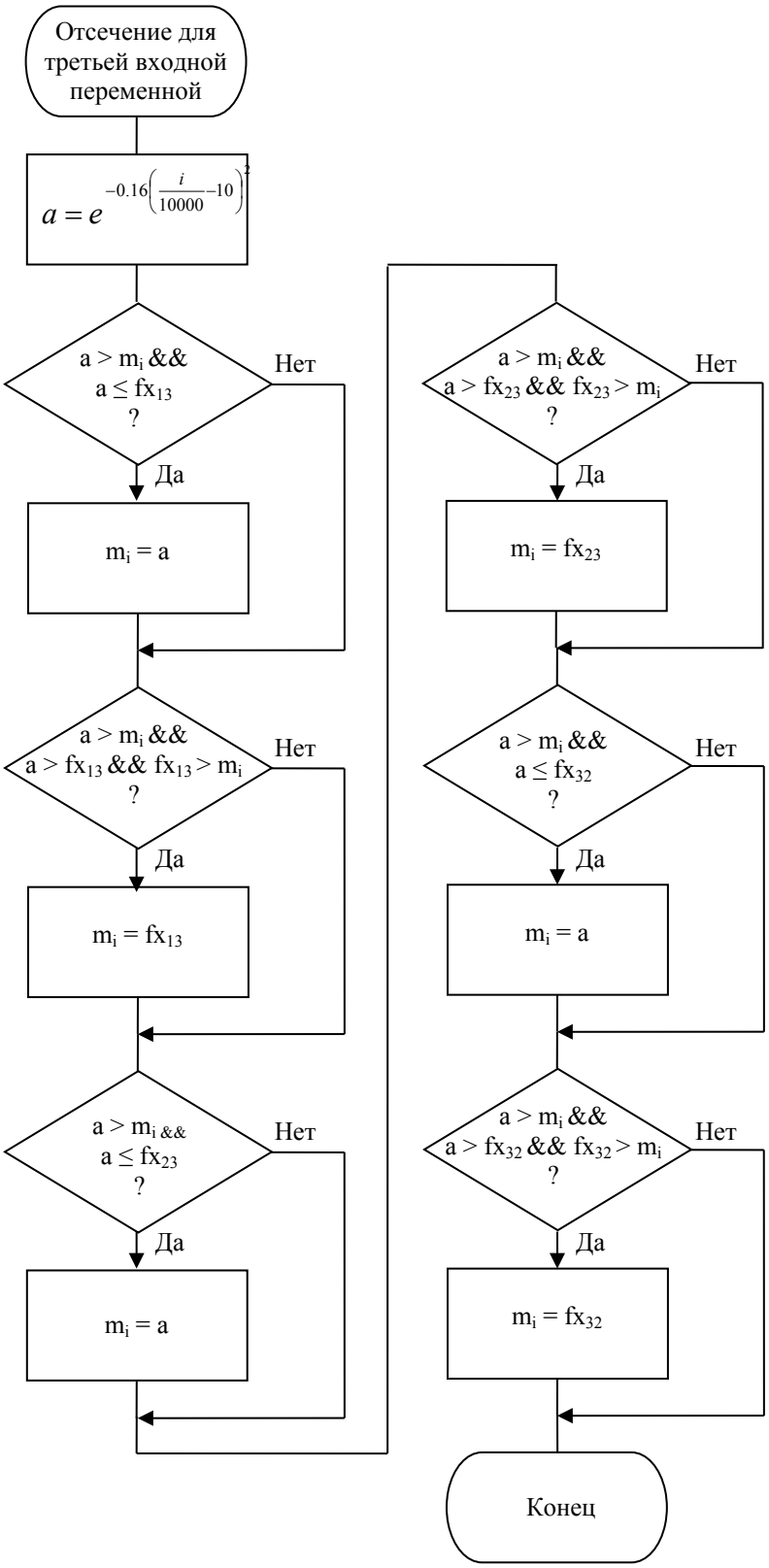

Проверка - моделирование нейронной сети
Ввод образа: s(4)(5)
p2(0) = p(0) p2(1) = p(1)
p2(2) = p(2) p2(3) = p(3)
kk = 0
kk < ns ?
Нет
Да
p(0) = p2(0) p(1) = p2(1)
p(2) = p2(2) p(3) = p2(3)
i = 0
Res(p(0)*8+p(1)*4+p(2)*2+(p3))+=1
kk = kk + 1
i = 0
Нет
i < 5 ?
Да
j = 0
i = i + 1
j < u(i) ?
Нет
Да
j = j + 1
k = rand()%4
p(k) = s(k)(i)
i = 0
i < 16 ?
Нет
Да
Res(i) = Res(i) / 16
i = i + 1
Подсчёт и вывод результата
Конец

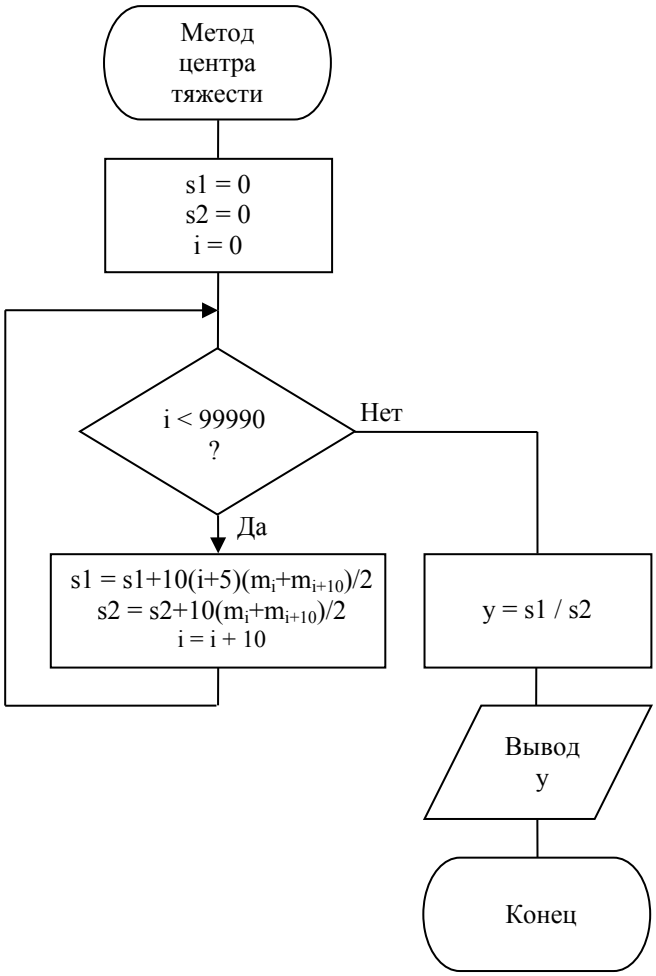

Подсчёт и вывод
tt(0)(i)=t(0)(i);
tt(1)(i)=t(1)(i)+t(2)(i)+t(4)(i)+t(8)(i);
tt(2)(i)=t(3)(i)+t(5)(i)+t(6)(i)+t(9)(i)
tt(2)(i)=tt(2)(i)+t(10)(i)+t(12)(i);
tt(3)(i)=t(7)(i)+t(11)(i)+t(13)(i)+t(14)(i);
tt(4)(i)=t(15)(i);
mine = 0
answer = 0
i = 0
i < 10 ?
Нет
Да
Вывод mine
Конец
e = 0
j = 0
j < 5 ?
Нет
Да
e(i) = e(i) + (tt(j)(i)-tt2(j))²
j = j + 1
e = e / 5
i = 0 ?
Нет
Да
e < mine ?
Нет
Да
mine = e
answer = 0
mine = e
answer = i
i = i + 1

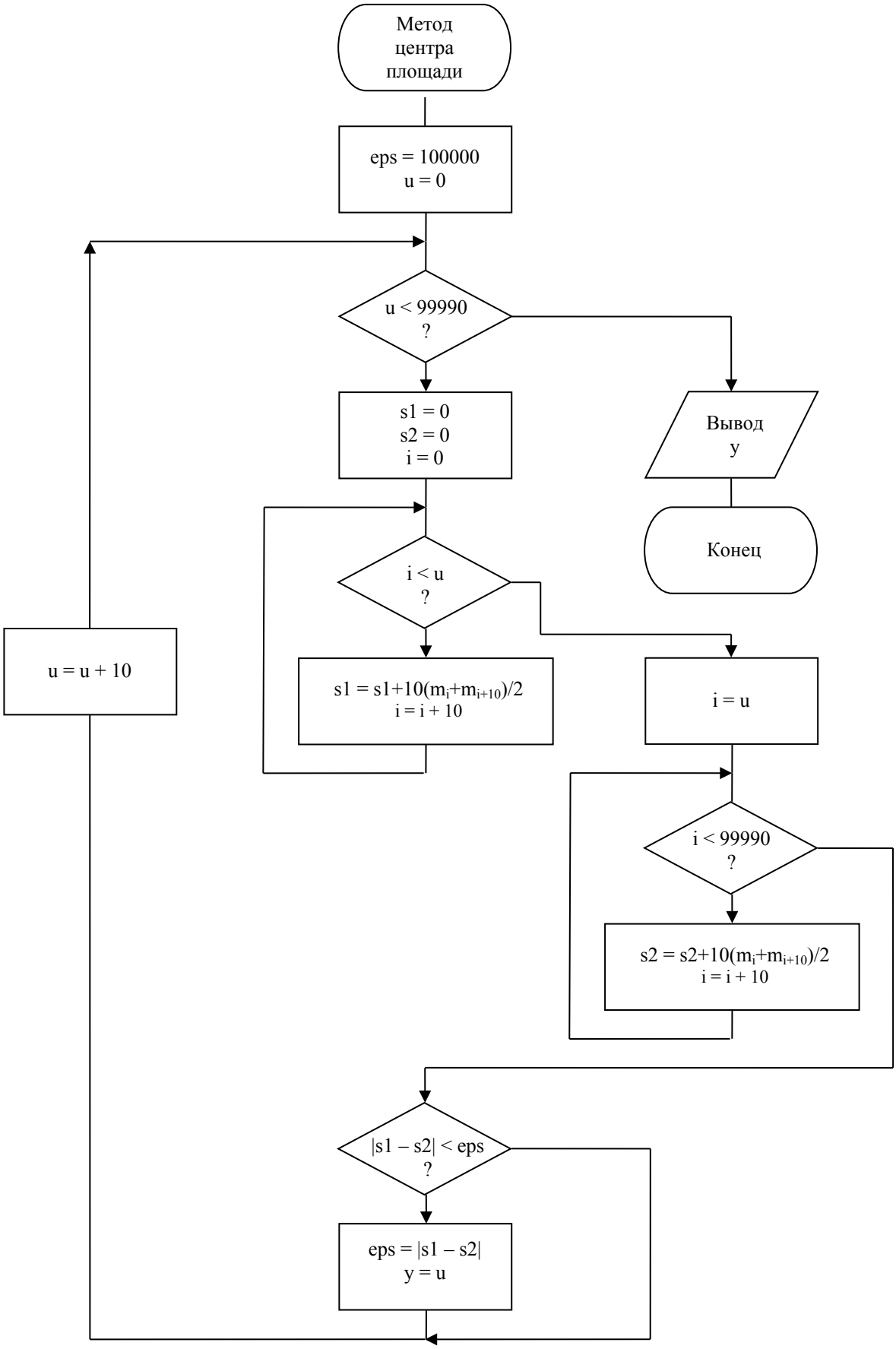
Блок-схема алгоритма для Лабораторной работы №7




Отсечение для первой входной переменной
$m_i = 0$
$a = e^{-0.16\left(\frac{i}{10000}\right)^2}$
a > m_i && a ≤ fx_11 ?
Нет
Да
m_i = a
a > m_i && a > fx_11 && fx_11 > m_i ?
Нет
Да
m_i = fx_11
a > m_i && a ≤ fx_21 ?
Нет
Да
m_i = a
a > m_i && a > fx_21 && fx_21 > m_i ?
Нет
Да
m_i = fx_21
a > m_i && a ≤ fx_31 ?
Нет
Да
m_i = a
a > m_i && a > fx_31 && fx_31 > m_i ?
Нет
Да
m_i = fx_31
Конец
Отсечение для второй входной переменной
$a = e^{-0.16\left(\frac{i}{10000}-5\right)^2}$
a > m_i && a ≤ fx_12 ?
Да
m_i = a
a > m_i && a > fx_12 && fx_12 > m_i ?
Да
m_i = fx_12
a > m_i && a ≤ fx_22 ?
Да
m_i = a
a > m_i && a > fx_22 && fx_22 > m_i ?
Да
m_i = fx_22
Конец


Отсечение для третьей входной переменной
$a = e^{-0.16\left(\frac{i}{10000}-10\right)^2}$
a > mi && a ≤ fx13 ?
Нет
Да
mi = a
a > mi && a > fx13 && fx13 > mi ?
Нет
Да
mi = fx13
a > mi && a ≤ fx23 ?
Нет
Да
mi = a
a > mi && a > fx23 && fx23 > mi ?
Нет
Да
mi = fx23
a > mi && a ≤ fx32 ?
Нет
Да
mi = a
a > mi && a > fx32 && fx32 > mi ?
Нет
Да
mi = fx32
Конец


Метод
центра
тяжести
s1 = 0
s2 = 0
i = 0
i < 99990
?
Нет
Да
s1 = s1+10(i+5)(m_i+m_i+10)/2
s2 = s2+10(m_i+m_i+10)/2
i = i + 10
y = s1 / s2
Вывод
y
Конец


Метод центра площади
eps = 100000
u = 0
u < 99990 ?
s1 = 0
s2 = 0
i = 0
Вывод y
Конец
i < u ?
s1 = s1+10(mi+mi+10)/2
i = i + 10
i = u
u = u + 10
i < 99990 ?
s2 = s2+10(mi+mi+10)/2
i = i + 10
|s1 – s2| < eps ?
eps = |s1 – s2|
y = u

Танцов Петр Николаевич
доц. каф. ЭИС МГГУ

ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Лабораторный практикум
для студентов направления подготовки 230201

Сдано в набор 30.05.13. Подписано к печати 30.05.13.
Формат 60x90/16. Бумага тип №1. Высокая печать.

Тираж 100 экз.