Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Курский государственный университет»

Кафедра программного обеспечения и администрирования

информационных систем

Направление подготовки математическое обеспечение и администрирование

информационных систем

Форма обучения очная

**Отчет**

**по лабораторной работе № 8**

«Многослойные нейронные сети»

дисциплина «Основы теории нейронных сетей»

Выполнила:

студентка группы 413 Мусонда Салиму

Проверил:

Профессор кафедры ПОиАИС Добрица В. П.

Курск, 2020

Вариант 3

Задача 1. Составить блок схему и обучающую программу двухслойной нейронной сети, интерпретирующей временной ряд, по алгоритму обратного распространения ошибки. В сети 6 входных и один выходной нейроны, в скрытом слое число нейронов взять не более 10. Параметры обучающей программы: шаг обучения α (0,1; 0,01), предполагаемая точность E = 0,01, временной шаг ∆t = 0,01, начальный момент t0 = 0, длина обучающей серии T = 1000. Значения временного ряда вычисляются по данной временной функции. Функция:

Задача 2. Провести обучение той же самой нейронной сети для того же временного ряда с адаптивным шагом обучения. Сравнить время обучения сетей в этих задачах с постоянным шагом обучения и с адаптивным шагом обучения.

Функция активации нейронов скрытого слоя:

Функция активации нейрона выходного слоя:

Формулы:

Среднеквадратичная ошибка, где L – количество входных образов, – эталонное значение, а - вычисленное:

Выход нейрона скрытого слоя, где n – количество нейронов входного слоя, в данном случае n = 6:

Выход сети, где h – количество нейронов скрытого слоя, в данном случае h = 10:

Изменение весовых коэффициентов и смещения выходного слоя:

Где под понимается значения весовых коэффициентов и смещения на предыдущем шаге.

Изменение весовых коэффициентов и смещений скрытого слоя:

Где под понимается значения весовых коэффициентов и смещений на предыдущем шаге.

Ошибка нейрона выходного слоя:

, где – эталонное значение, а - вычисленное

Взвешенная сумма нейрона выходного слоя, где h – количество нейронов скрытого слоя, в данном случае h = 10:

Взвешенная сумма j-го нейрона скрытого слоя, где n – количество нейронов входного слоя, в данном случае n = 6:

Ошибка i-го нейрона скрытого слоя:

Изменение весовых коэффициентов и смещений нейронной сети происходит по следующим формулам:

Где под понимается значения весовых коэффициентов и смещений на предыдущем шаге.

Адаптивный шаг обучения для скрытого слоя:

Адаптивный шаг обучения для выходного слоя:

В качестве входных значений сначала выбираются элементы временного ряда с 0 по 5, затем с 1 по 6 и т. д. последнее от 988 до 993. Последовательность от 994 до 999 используется для проверки в качестве входных данных, а вычисленное 1000-ое значение сравнивается с 1000-ым значением временного ряда для доказательства обученности сети. Так как все входные значения должны лежать в промежутке от 0 до 1, каждое входное значение делится на одно и то же число. В нашем случае на 21, так как < 21 для любого t от 0 до 100.

Блок-схема алгоритма представлена на рисунках 1 - 4.

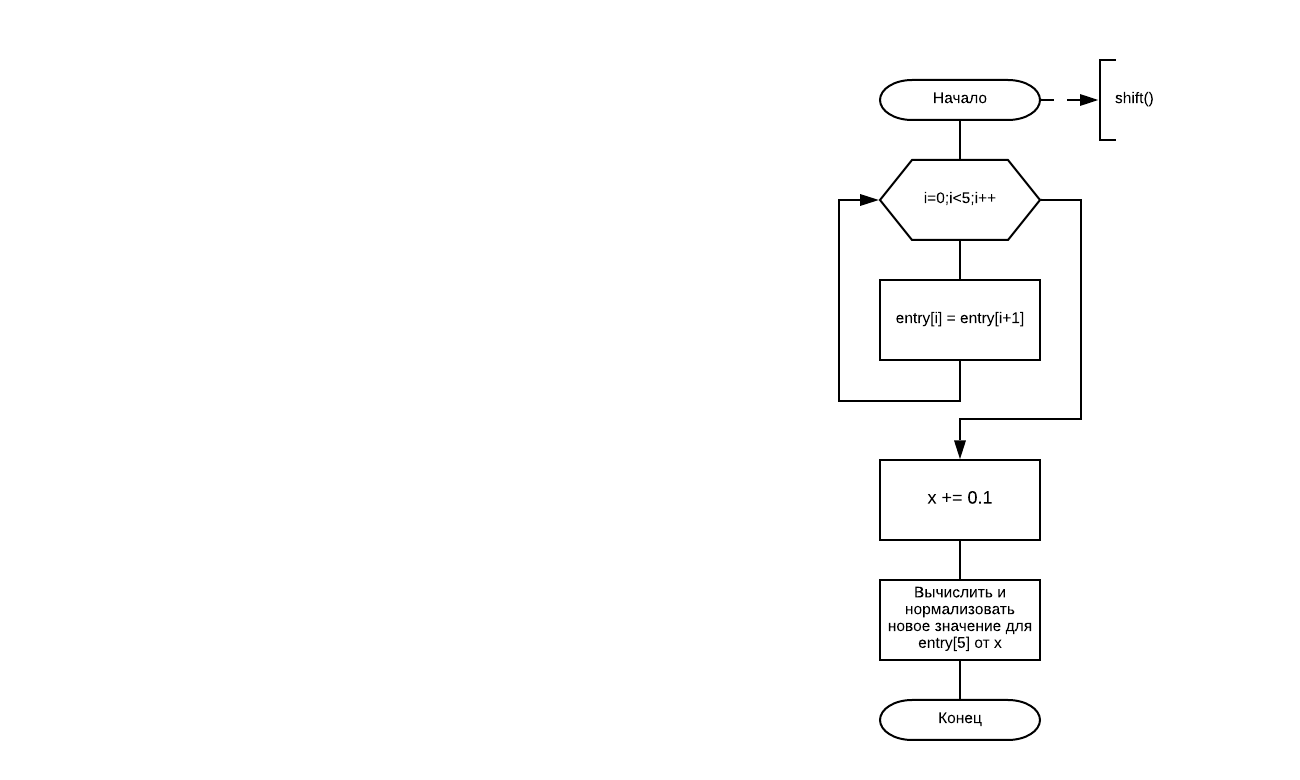


Рисунок 1 – Блок-схема функции shift(), сдвигающей массив входных значений entry на единицу вправо

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Блок-схема функции init(), инициализирующей значения входных данных

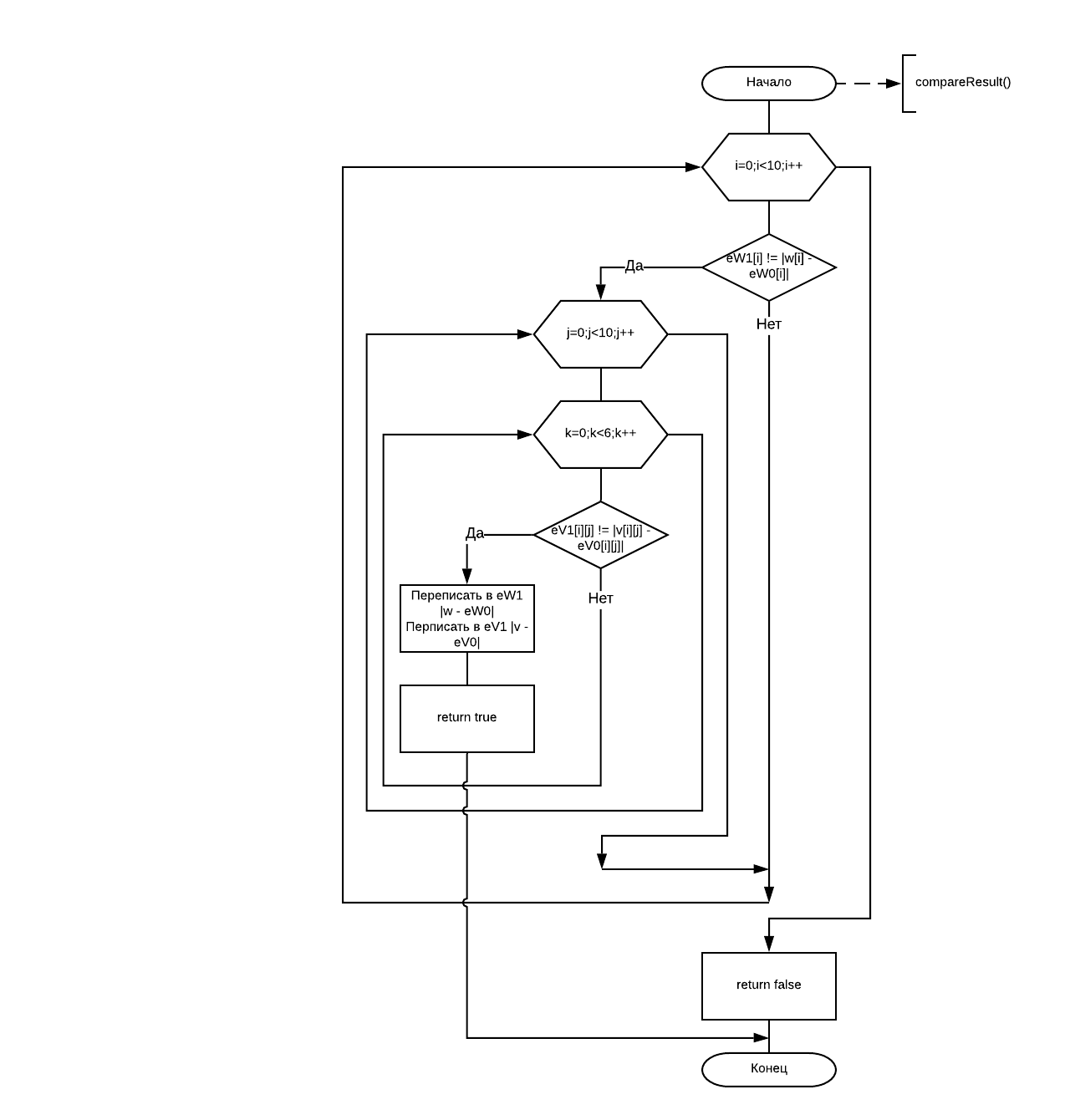


Рисунок 3 – Блок-схема функции compareResult(), проверяющая изменение результатов

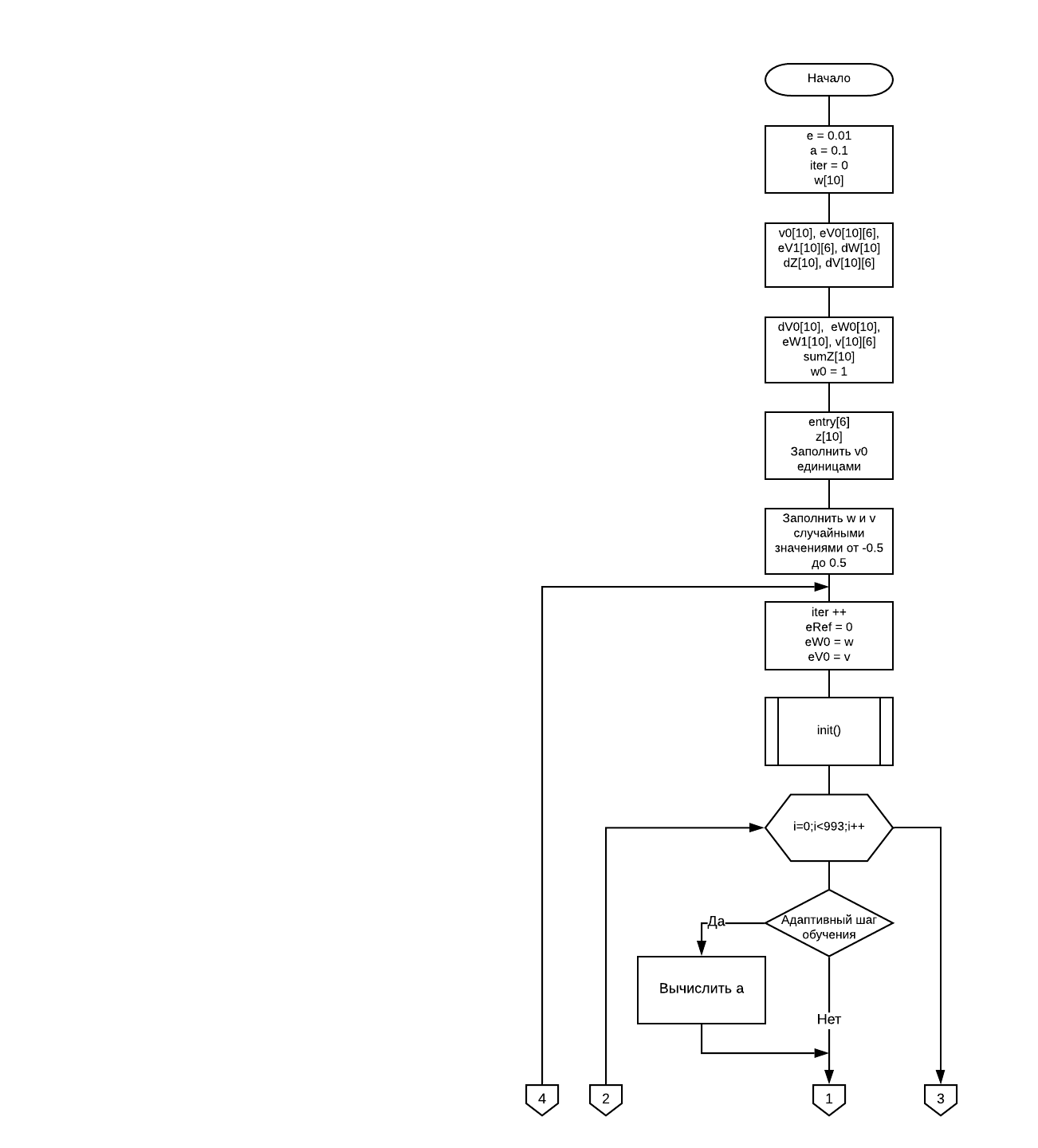


Рисунок 4 – Блок-схема главной функции

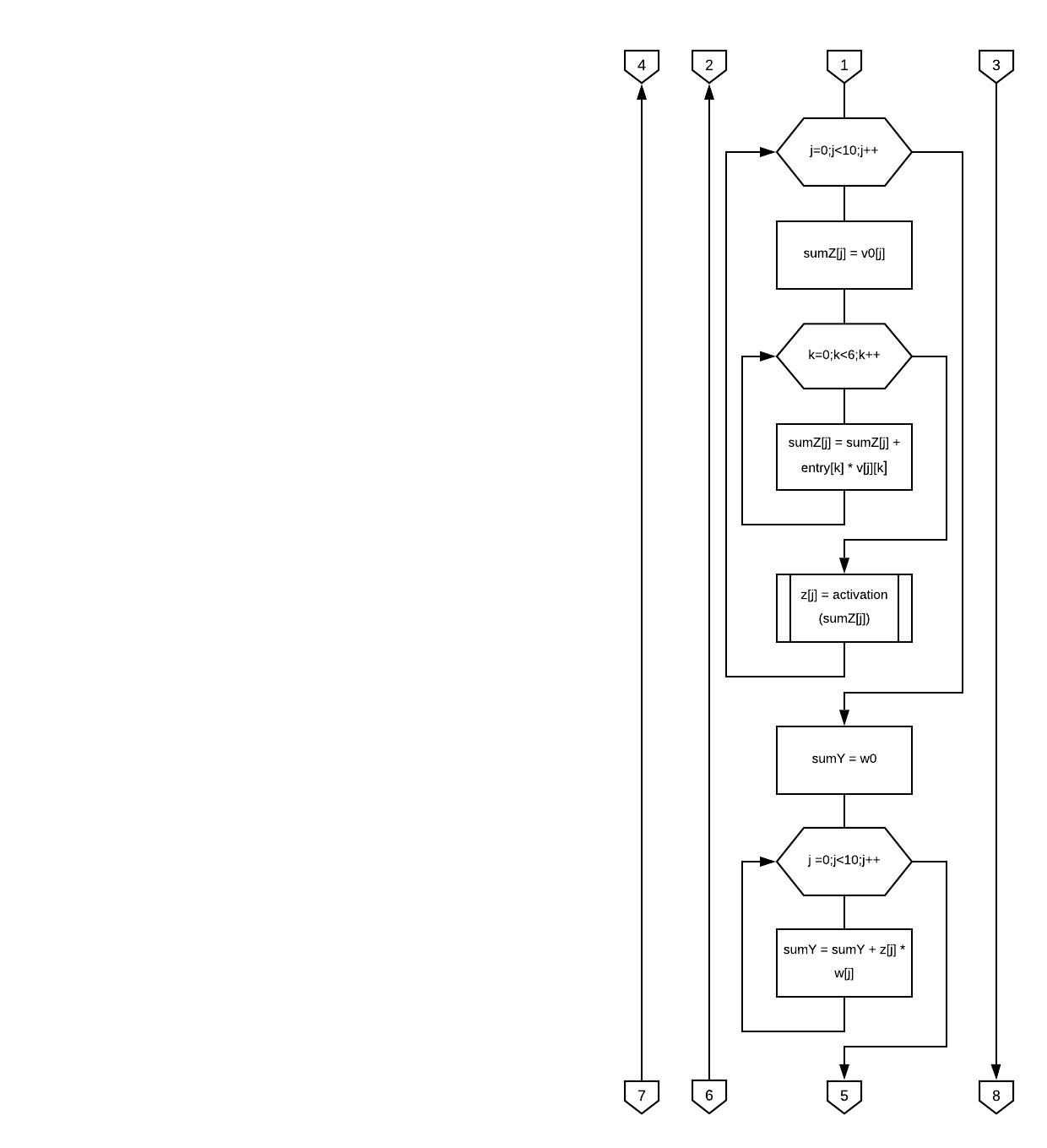


Рисунок 4 – Продолжение

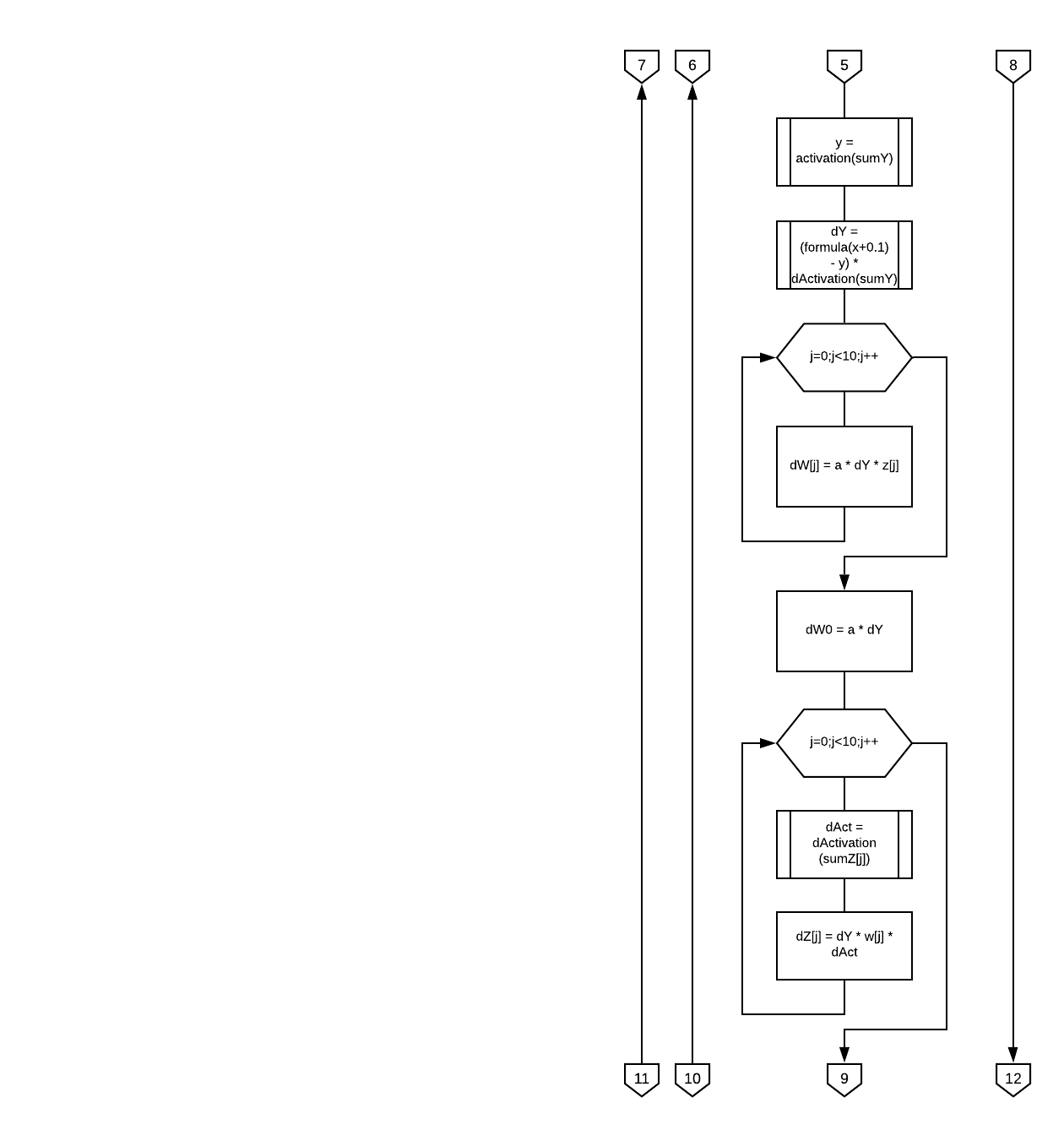


Рисунок 4 – Продолжение

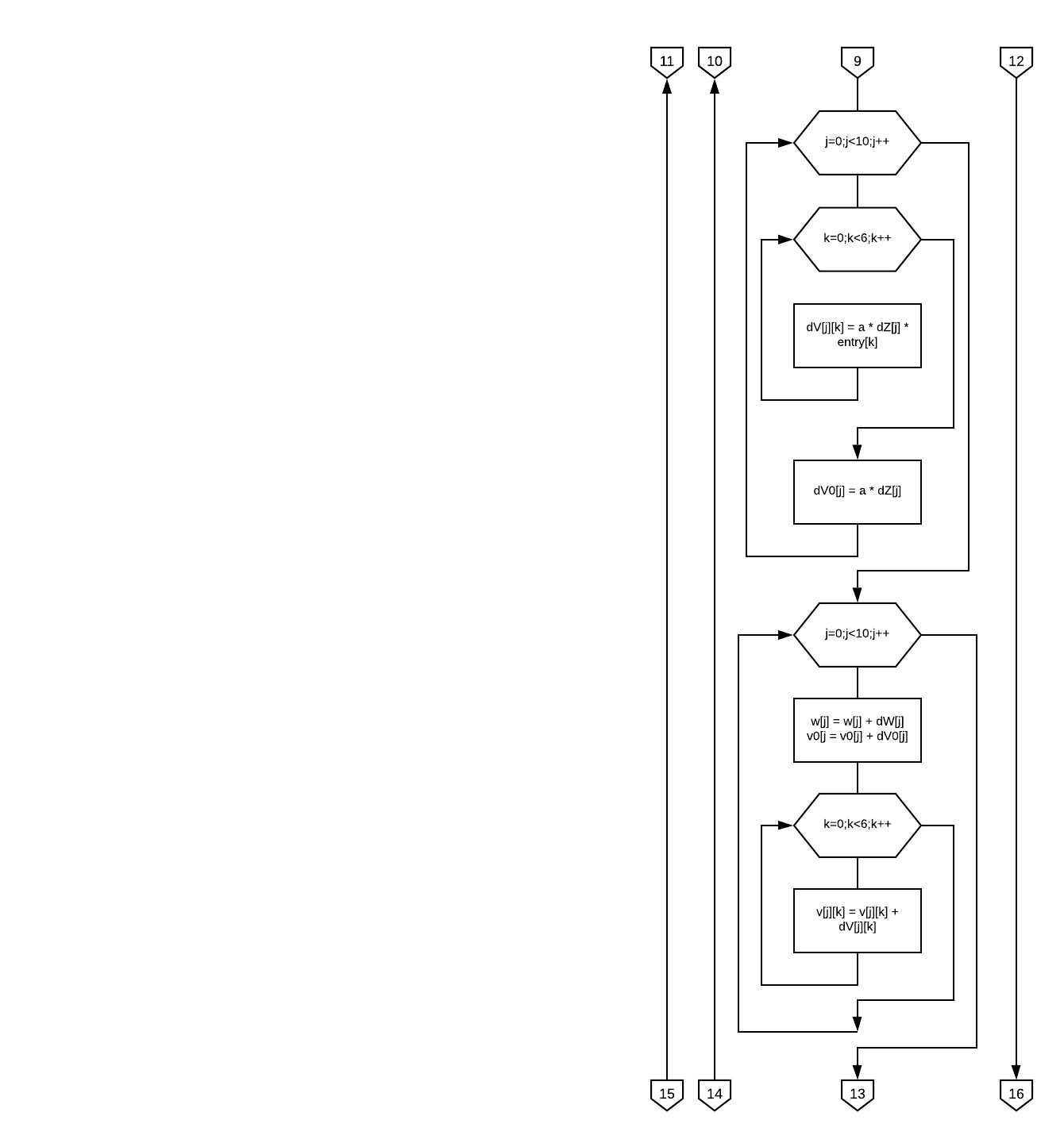


Рисунок 4 – Продолжение

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Продолжение

Текст программы

**import** javax.swing.\*;  
**import** javax.swing.event.TableModelEvent;  
**import** javax.swing.event.TableModelListener;  
**import** javax.swing.table.DefaultTableModel;  
**import** javax.swing.table.TableModel;  
**import** java.awt.\*;  
**import** java.awt.event.\*;  
  
**class** MForm {  
 JFrame **mframe1**;  
 JRadioButton **jrbA**;  
 JButton **jbStudy**;  
 MCanvas **canvas**;  
 JLabel **jlForResult**;  
 JLabel **jlForIdeal**;  
 JLabel **jlForIter**;  
 JLabel **jlResult**;  
 JLabel **jlIdeal**;  
 JLabel **jlIter**;  
  
 **public** MForm() {  
 **mframe1** = **new** JFrame(**"Окно для ввода"**);  
 **mframe1**.setSize(400, 450);  
 **mframe1**.setDefaultCloseOperation(WindowConstants.***EXIT\_ON\_CLOSE***);  
 **canvas** = **new** MCanvas();  
 **mframe1**.add(**canvas**);  
 **jrbA** = **new** JRadioButton(**"Адаптивный шаг обучения"**);  
 **jrbA**.setLocation(10, 10);  
 **jrbA**.setSize(250, 30);  
 **jlForIdeal** = **new** JLabel(**"Эталонное значение: "**);  
 **jlForIdeal**.setSize(250,25);  
 **jlForIdeal**.setLocation(10, 120);  
 **jlIdeal** = **new** JLabel();  
 **jlIdeal**.setLocation(150, 120);  
 **jlIdeal**.setSize(100, 25);  
 **jlForResult** = **new** JLabel(**"Вычисленное значение: "**);  
 **jlForResult**.setLocation(10, 155);  
 **jlForResult**.setSize(250, 25);  
 **jlResult** = **new** JLabel();  
 **jlResult**.setSize(100, 25);  
 **jlResult**.setLocation(160, 155);  
 **jlForIter** = **new** JLabel(**"Количество итераций: "**);  
 **jlForIter**.setLocation(10, 190);  
 **jlForIter**.setSize(250, 25);  
 **jlIter** = **new** JLabel();  
 **jlIter**.setSize(100, 25);  
 **jlIter**.setLocation(150, 190);  
 **jbStudy** = **new** JButton(**"Начать обучение"**);  
 **jbStudy**.setLocation(10, 70);  
 **jbStudy**.setSize(250, 30);  
 **jbStudy**.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 Network network = **new** Network(**jbStudy**.isSelected());  
 **if**(network.study()) {  
 network.check(**jlResult**, **jlIdeal**, **jlIter**);  
 }  
 **else** {  
 **jlResult**.setText(**"Обучение невозможно"**);  
 }  
 }  
 });  
  
 **canvas**.add(**jlIter**);  
 **canvas**.add(**jlForIter**);  
 **canvas**.add(**jlForIdeal**);  
 **canvas**.add(**jlForResult**);  
 **canvas**.add(**jlResult**);  
 **canvas**.add(**jlIdeal**);  
 **canvas**.add(**jrbA**);  
 **canvas**.add(**jbStudy**);  
 **mframe1**.setVisible(**true**);  
 }  
  
 **public static void** main(String[] args) {  
 MForm mForm = **new** MForm();  
  
 }  
}

**import** javax.swing.\*;  
  
**public class** Network {  
  
 **private double x**, **w0**, **e**, **a**, **y**, **sumY**, **dY**, **dW0**;  
  
 **private double** []**w**, **dW**, **z**, **entry**, **v0**, **sumZ**, **dZ**, **dV0**, **eW0**, **eW1**;  
  
 **private double** [][] **v**, **dV**, **eV0**, **eV1**;  
  
 **int iter**;  
  
 **private boolean adapt**;  
  
 **public** Network(**boolean** apapt) {  
 **this**.**adapt** = **adapt**;  
 **e** = 0.01;  
 **iter** = 0;  
 **a** = 0.1;  
 **eW0** = **new double**[10];  
 **eW1** = **new double**[10];  
 **eV0** = **new double**[10][6];  
 **eV1** = **new double**[10][6];  
 **w** = **new double**[10];  
 **v0** = **new double**[10];  
 **dW** = **new double**[10];  
 **dZ** = **new double**[10];  
 **dV** = **new double**[10][6];  
 **dV0** = **new double**[10];  
 **for**(**int** i = 0; i < 10; i ++) {  
 **w**[i] = Math.*random*() - 0.5;  
 }  
 **v** = **new double**[10][6];  
 **for**(**int** i = 0; i < 10; i ++) {  
 **for**(**int** j = 0; j < 6; j ++) {  
 **v**[i][j] = Math.*random*() - 0.5;  
 **v0**[j] = 1;  
 }  
 }  
 **sumZ** = **new double**[10];  
 **w0** = 1;  
 **entry** = **new double**[6];  
 **z** = **new double**[10];  
 }  
  
 **private double** formula(**double** t) {  
 **return** ((t / 5 + Math.*sin*(t + 2))/21);  
 }  
  
 **private void** init() {  
 **x** = 0;  
 **for**(**int** i = 0; i < 6; i ++) {  
 **entry**[i] = formula(**x**);  
 **x** += 0.1;  
 }  
 }  
  
 **private double** activation(**double** x) {  
 **return** (1.0 / (1 + Math.*exp*(-x)));  
 }  
  
 **private double** dActivation(**double** x) {  
 **return** (Math.*exp*(-x) / Math.*pow*((1 + Math.*exp*(-x)), 2));  
 }  
  
 **private void** shift() {  
 **for**(**int** i = 0; i < 5; i ++) {  
 **entry**[i] = **entry**[i+1];  
 }  
 **x** += 0.1;  
 **entry**[5] = formula(**x**);  
 }  
  
  
 **private boolean** compareResult() {  
 **for**(**int** i = 0; i < 10; i ++) {  
 **if**(**eW1**[i] != Math.*abs*(**w**[i] - **eW0**[i])) {  
 rewriteEW1();  
 **for**(**int** j = 0; j < 10; j ++) {  
 **for**(**int** k = 0; k < 6; k ++) {  
 **if**(**eV1**[i][j] != Math.*abs*(**v**[i][j] - **eV0**[i][j])) {  
 rewriteEv1();  
 **return true**;  
 }  
 }  
 }  
 }  
 }  
 **return false**;  
 }  
  
 **private void** rewriteEW1() {  
 **for**(**int** i = 0; i < 10; i ++) {  
 **eW1**[i] = Math.*abs*(**w**[i] - **eW0**[i]);  
 }  
 }  
  
 **private void** rewriteEv1() {  
 **for**(**int** i = 0; i < 10; i ++) {  
 **for**(**int** j = 0; j < 6; j ++) {  
 **eV1**[i][j] = Math.*abs*(**v**[i][j] - **eV0**[i][j]);  
 }  
 }  
 }  
  
 **private void** calculateA(int []mas) {  
 **double** sum = 0;  
 **for**(**int** i = 0; i < 6; i ++) {  
 sum += **mas**[i] \* **mas**[i];  
 }  
 **a** = 1.0 / (1 + sum);  
 }  
  
 **public void** calculateY() {  
 **for** (**int** j = 0; j < 10; j++) {  
 **sumZ**[j] = **v0**[j];  
 **for** (**int** k = 0; k < 6; k++) {  
 **sumZ**[j] += **entry**[k] \* **v**[j][k];  
 }  
 **z**[j] = activation(**sumZ**[j]);  
 }  
 **sumY** = **w0**;  
 **for** (**int** j = 0; j < 10; j++) {  
 **sumY** += **z**[j] \* **w**[j];  
 }  
 **y** = activation(**sumY**);  
 }  
  
 **public void** rewrite() {  
 **for**(**int** i = 0; i < 10; i ++) {  
 **eW0**[i] = **w**[i];  
 **for**(**int** j = 0; j < 6; j ++) {  
 **eV0**[i][j] = **v**[i][j];  
 }  
 }  
 }  
  
 **public boolean** study() {  
 **double** eRef;  
 **do** {  
 **iter**++;  
 eRef = 0;  
 rewrite();  
 init();  
 **for** (**int** i = 0; i < 993; i++) {  
 **if**(**adapt**) {  
 calculateA(z);  
 }  
 **for** (**int** j = 0; j < 10; j++) {  
 **sumZ**[j] = **v0**[j];  
 **for** (**int** k = 0; k < 6; k++) {  
 **sumZ**[j] += **entry**[k] \* **v**[j][k];  
 }  
 **z**[j] = activation(**sumZ**[j]);  
 }  
 **sumY** = **w0**;  
 **for** (**int** j = 0; j < 10; j++) {  
 **sumY** += **z**[j] \* **w**[j];  
 }  
 **y** = activation(**sumY**);  
 **dY** = (formula((**x** + 0.1)) - **y**) \* dActivation(**sumY**);  
 **for** (**int** j = 0; j < 10; j++) {  
 **dW**[j] = **a** \* **dY** \* **z**[j];  
 }  
 **dW0** = **a** \* **dY**;  
 **for** (**int** j = 0; j < 10; j++) {  
 **dZ**[j] = **dY** \* **w**[j] \* dActivation(**sumZ**[j]);  
 }

**if**(**adapt**) {  
 calculateA(entry);  
 }  
 **for** (**int** j = 0; j < 10; j++) {  
 **for** (**int** k = 0; k < 6; k++) {  
 **dV**[j][k] = **a** \* **dZ**[j] \* **entry**[k];  
 }  
 **dV0**[j] = **a** \* **dZ**[j];  
 }  
 **for** (**int** j = 0; j < 10; j++) {  
 **w**[j] = **w**[j] + **dW**[j];  
 **v0**[j] = **v0**[j] + **dV0**[j];  
 **for** (**int** k = 0; k < 6; k++) {  
 **v**[j][k] = **v**[j][k] + **dV**[j][k];  
 }  
 }  
 **w0** = **w0** + **dW0**;  
 shift();  
 eRef += 0.5 \* Math.*pow*(**entry**[5] - **y**,2);  
 }  
 **if**(!compareResult()) {  
 **return false**;  
 }  
 }**while**(eRef > **e**);  
 **return true**;  
}

**public void** check(JLabel jlResult, JLabel jlIdeal, JLabel jlIter) {  
 calculateY();  
 jlResult.setText(String.*valueOf*(**y**));  
 shift();  
 jlIdeal.setText(String.*valueOf*(**entry**[5]));  
 jlIter.setText(String.*valueOf*(**iter**));  
 }  
}

**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.util.ArrayList;  
  
**public class** MCanvas **extends** JComponent {  
  
 MCanvas() {  
  
 }  
  
 **public void** paintComponent(Graphics g) {  
 **super**.paintComponents(g);  
 }  
}

Результаты работы программы представлены на рисунке 5 - 6.

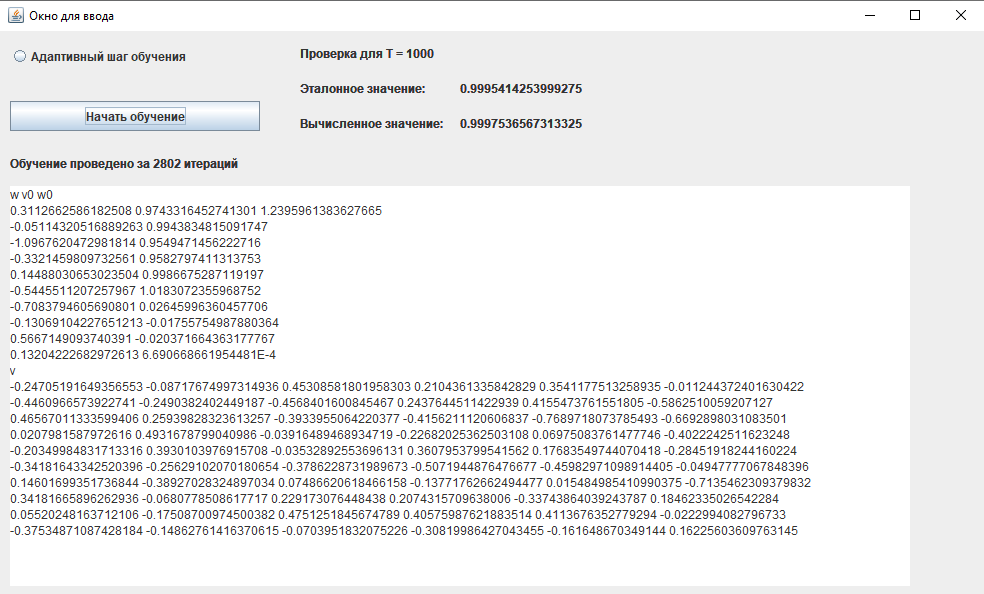
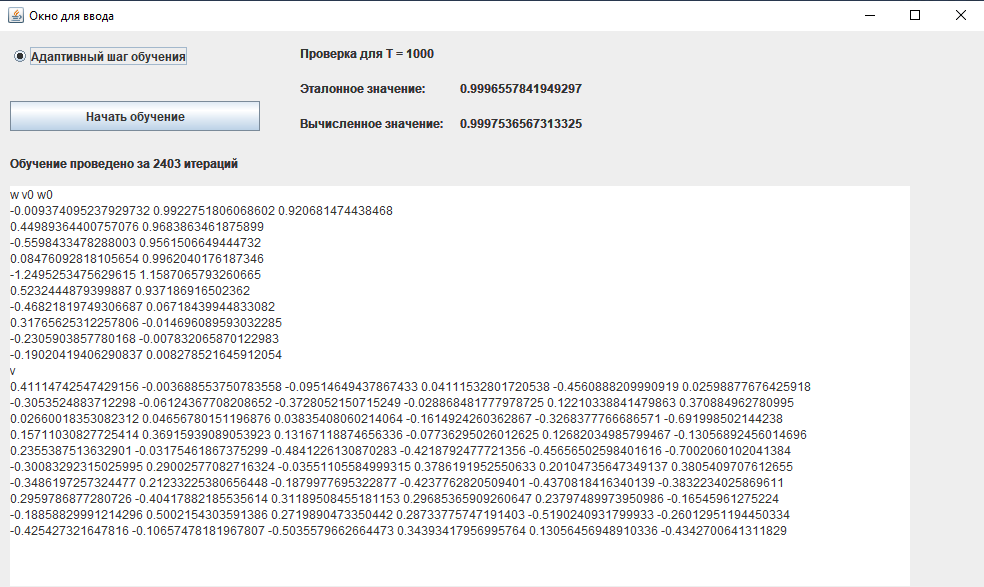


Рисунок 5 – Результат работы программы без адаптивного шага обучения

Рисунок 6 – Результат работы программы с адаптивным шагом обучения

Вывод: Адаптивный шаг обучения позволяет повысить скорость и точность обучения.