**КОНТРОЛЬНАЯ РАБОТА**

По дисциплине: Интеллектуальные информационные системы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

на тему: Искусственные нейронные сети на основе алгоритма обратного распространения ошибки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность: Прикладная информатика

(в экономике)\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Группа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работу проверил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2014

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc355462349)

[1. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ НЕЙРОННОЙ СЕТИ С АЛГОРИТМОМ ОБРАТНОГО РАСПРОСТАНЕНИЯ ОШИБКИ 4](#_Toc355462350)

[2. РАСЧЕТ НЕЙРОННОЙ СЕТИ](#_Toc355462350) 6

[3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ НЕЙРОННОЙ СЕТИ](#_Toc355462351) 8

[3.1 Общие сведения](#_Toc355462352) 8

[3.2 Функциональное назначение](#_Toc355462353) 9

[3.3 Описание логической структуры](#_Toc355462354) 9

[3.4 Требования к техническому обеспечению.](#_Toc355462355) 13

[3.5 Вызов программы.](#_Toc355462355) 13

[3.6 Входные данные.](#_Toc355462355) 13

[3.7 Выходные данные.](#_Toc355462355) 14

[3.8 Описание тестовых прогонов.](#_Toc355462355) 14

[ЗАКЛЮЧЕНИЕ](#_Toc355462358) 15

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_Toc355462359) 16

[ПРИЛОЖЕНИЕ. КОД ПРОГРАММЫ](#_Toc355462359) 17

ВВЕДЕНИЕ

В современном мире все чаще мы сталкиваемся с проблемами, которые подчас решить может только человек — ЭВМ с ними справиться не в силах. Примером таких задач могут служить задачи прогнозирования (экономического, климатического), задачи классификации объектов, задачи распознавания символов и т.п. Для решения таких задач используются т.н. искусственные нейронные сети (ИНС) — тривиальные модели функционирования человеческого головного мозга. На данный момент существует огромное количество видов ИНС, которые различаются поставленными перед ними задачами, физической структурой сети, внутренними параметрами: функцией активации, алгоритмом обучения и т.д. В данной работе нами была построена и изучена ИНС с одним скрытым слоем, с алгоритмом обучения по методу обратного распространения ошибки. Раздел 1 содержит математическую модель моделируемой нами сети. В разделе 2 описаны расчеты ИНС, в соответствии с вариантом работы. Раздел 3 содержит описание разработанной нами программы на ЭВМ, которая инициализирует ИНС с начальными параметрами, обучает ее и тестирует на некотором наборе входных данных.

1. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ НЕЙРОННОЙ СЕТИ С АЛГОРИТМОМ ОБРАТНОГО РАСПРОСТАНЕНИЯ ОШИБКИ

В соответствие с вариантом работы нам необходимо построить ИНС со структурой 2-2-1, то есть два входных нейрона, два нейрона в скрытом слое и один выходной нейрон. Также необходимо учесть два нейрона смещения, один из которых находиться на входном слое, второй на скрытом слое. Схематически рассматриваемую сеть изобразим следующим образом

Схема 1. Структура ИНС 2-2-1.

В виду того, что ожидаемым откликом от сети, в соответствии с нашим вариантом, является сигнал , то в качестве функции активации для нейронов на скрытом слое и для выходного нейрона выберем функцию

Опишем математическую модель ИНС.

Взвешенная сумма нейрона

где суммирование ведется по всем номерам нейронов на предыдущем слое. Отклик нейрона

Ошибка выходного нейрона

Ошибка нейронов скрытого слоя

где . Изменение веса нейрона после этапа обучения

где — начальный вес ребра, соединяющего нейроны и , — вес ребра, соединяющего нейроны и на -ом шаге обучения, — норма обучения, — норма инерции обучения. В результате одного шага обучения веса ребер в ИНС изменяются по следующему правилу

Обучение ИНС проходит до тех пор, пока невязка выходного сигнала превышает допустимый порог

После того, как выходной сигнал ИНС по модулю не будет превышать ожидаемого выходного сигнала более чем на процесс обучения останавливается и сеть считается обученной.

2. РАСЧЕТ НЕЙРОННОЙ СЕТИ

В соответствие с вариантом работы начальные данные для ИНС

Коэффициенты и допуски выберем следующим образом

Обучение сети необходимо проводить до тех пор, пока не выполнится условие, описанное в разделе 1. Но мы ограничимся одним прямым и обратным проходом.

*(обучение следует продолжить)*

3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ НЕЙРОННОЙ СЕТИ

3.1 Общие сведения

С целью визуализации процесса обучения ИНС нами была разработана программа на ЭВМ, которая позволяет наглядно увидеть каждый шаг обучения ИНС по методу обратного распространения ошибки. Интерфейс программы представлен на рисунке 2. Назначение каждого отдельного элемента этого интерфейса мы опишем нижу. Сейчас же заметим, что в целом программа оформлена в виде ИНС типа 2-2-1.

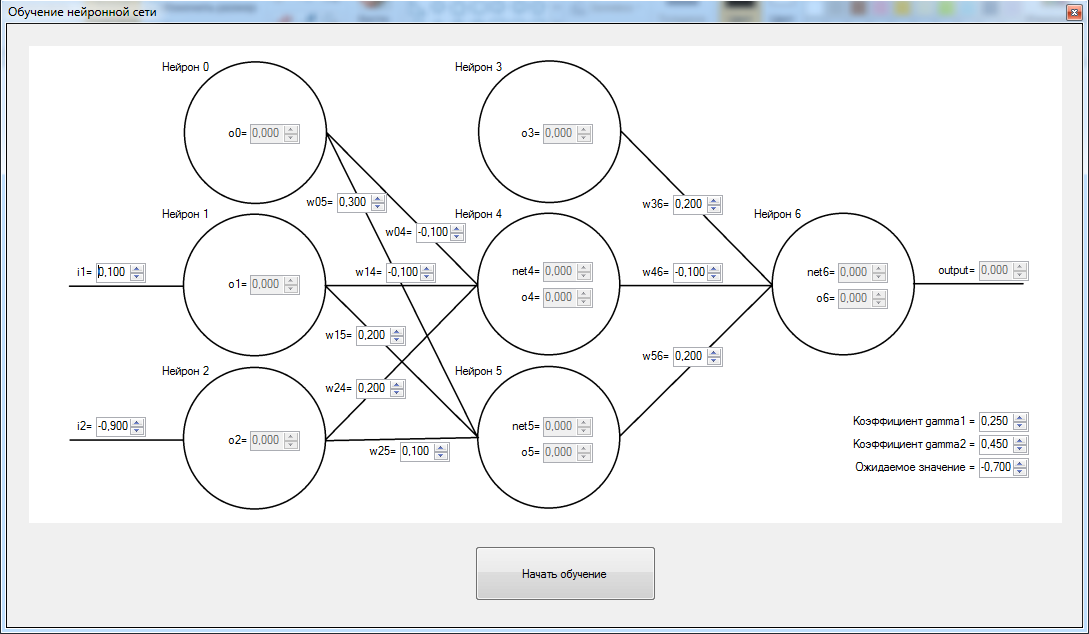


Рис 2. Интерфейс приложения.

3.2 Функциональное назначение

Данное приложение создано с целью обучения ИНС типа 2-2-1. Изначально, до обучения, ИНС представляет из себя структуру нейронов, которые, принимая входные сигналы, на выходе образуют сигнал, не имеющий никакого смысла. С помощью метода обратного распространения ошибки, шаг за шагом, при помощи данной программы мы можем обучить ИНС таким образом, чтобы выход сети начал выдавать нужную нам информацию, в частности сигнал, который мы от ИНС ожидаем.

3.3 Описание логической структуры

Алгоритм работы программы представлен на схеме 3.

step = 0

isFirstEra = 0

Ввод i1, i2

Ввод w04, w05

Ввод w14, w15

Ввод w24, w25

Ввод w36, w46, w56

Ввод gamma1, gamma2

Ввод th

step

step = 1

0

1

o0 = 1

o1 = i1

o2 = i2

Вывод o0

Вывод o1

Вывод o2

net4 = o0\*w04+o1\*w14+o2\*w24

o4 = tanh(net4)

net5 = o0\*w05+o1\*w15+o2\*w25

o5 = tanh(net5)

o3 = 1

2

step = 2

Вывод net4,o4

Вывод net5,o5

Вывод o3

step = 3

3

net6 = o3\*w36+o4\*w46+o5\*w56

o6 = tanh(net6)

Вывод net6,o6

step = 4

4

output = o6

|output – th| < 0.001

step = 5

Вывод “ИНС обучена!»

step = 6

5

6

delta6 = (th-o6)(o6+1)(1-o6)

delta5 = (o5+1)(1-o5)\*delta6\*w56

delta4 = (o4+1)(1-o4)\*delta6\*w46

isFirstEra = true

dw56 = gamma1\*delta6\*o5

dw46 = gamma1\*delta6\*o4

dw36 = gamma1\*delta6\*o3

dw56 = gamma1\*delta6\*o5+gamma2\*dw56

dw46 = gamma1\*delta6\*o4+gamma2\*dw46

dw36 = gamma1\*delta6\*o3+gamma2\*dw36

w56 = w56 + dw56

w46 = w46 + dw46

w36 = w36 + dw36

Вывод w56,w46,w36

step = 7

7

delta6 = (th-o6)(o6+1)(1-o6)

delta5 = (o5+1)(1-o5)\*delta6\*w56

delta4 = (o4+1)(1-o4)\*delta6\*w46

isFirstEra = true

dw25 = gamma1\*delta5\*o2

dw24 = gamma1\*delta4\*o2

dw15 = gamma1\*delta5\*o1

dw14 = gamma1\*delta4\*o1

dw05 = gamma1\*delta5\*o0

dw04 = gamma1\*delta4\*o0

dw25 = gamma1\*delta5\*o2+gamma2\*dw25

dw24 = gamma1\*delta4\*o2+gamma2\*dw24

dw15 = gamma1\*delta5\*o1+gamma2\*dw15

dw14 = gamma1\*delta4\*o1+gamma2\*dw14

dw05 = gamma1\*delta5\*o0+gamma2\*dw05

dw04 = gamma1\*delta4\*o0+gamma2\*dw04

w25 = w25 + dw25

w24 = w24 + dw24

w15 = w15 + dw15

w14 = w14 + dw14

w05 = w05 + dw05

w04 = w04 + dw04

Вывод w04,w14,w24

Вывод w05,w15,w25

isFirstEra = false

step = 2

Схема 3. Блок-схема приложения.

3.4 Требования к техническому обеспечению

Реализованное приложение создано в среде Visual Studio 2010 и использует .NetFramework v4.5. На клиентском ЭВМ должна быть установлена видеокарта, поддерживающая технологию OpenGL. Также должен быть установлен пакет .NetFramework версии 4.5. Так как приложение выполняет расчеты, то необходим хороший процессор, не ниже Intel Pentium 4, 1.8 GHz. Оперативная память также используется, поэтому необходимо от 128 Mb оперативной памяти на материнской плате.

3.5 Вызов программы

Программа имеет структуру приложения Win32, может быть вызвана из проводника двойным нажатие левой клавиши мыши на значке neuro.exe. Также программа может быть вызвана командой neuro.exe из консоли (без параметров).

3.6 Входные данные

Программа не принимает параметры из консоли. Ввод входных данных организован в диалоговом окне приложения непосредственно через элементы управления. Пользователь, перед началом обучения ИНС, может изменить входные сигналы сети, начальные синаптические веса, коэффициенты обучения и ожидаемый отклик. Все эти параметры, до начала процесса обучения, доступны пользователю. После начала процесса обучения все поля становятся недоступными для редактирования, так как программа начинает изменять эти поля в соответствии с запрограммированным алгоритмом.

3.7 Выходные данные

После некоторого количества шагов ИНС обучается и выдает на выход сигнал, который пользователь ожидает от нее. В качестве результата работы программы пользователь имеет обученные значения синаптических весов, которые он может использовать в дальнейшем для практических приложений данной ИНС.

3.8 Описание тестовых прогонов

При инициализации приложения все необходимые поля заполнены данными по умолчанию, которые соответствуют варианту задания. Пользователь, изменяя значение в числовых полях, может изменять все имеющиеся данные и коэффициенты по своему усмотрению. После выставления необходимых параметрах пользователю следует нажать кнопку «Начать обучение», после чего эта кнопка меняет название на «Далее». Процесс обучения ИНС запущен и теперь пользователю уже нельзя менять поля с данными. У пользователя есть одна единственная возможность — нажимать кнопку «Далее» и наблюдать за прямыми и обратными проходами по алгоритму обратного распространения ошибки применительно к ИНС, которую он задал. После того, как выходной сигнал становиться похожим на ожидаемый сигнал, пользователю выдается сообщение «ИНС обучена!». Работа программы на этом завершается. Результаты тестовых прогонов приложения при разных параметрах представлены в таблице 4.



Таблица 4. Результаты тестовых прогонов приложения.

ЗАКЛЮЧЕНИЕ

В данной работе была построена математическая модель ИНС типа 2-2-1, рассчитаны веса ребер сети, написано приложение для ЭВМ, имитирующее процесс обучения ИНС. Данную ИНС в дальнейшем можно доработать, усложнив модель, и использовать для решения прикладных задач.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Круглов В.В., Борисов В.В. Искусственные нейронные сети. Теория и практика. — М.: Горячая линия – Телеком, 2001. 382 с.
2. Горбань А.Н., Россиев Д.А. Нейронные сети на персональном компьютере. — Новосибирск: Наука, 1996. 275 с.
3. Р. Тадеусевич, Б. Боровик, Т. Гончаж, Б. Леппер, Элементарное введение в технологию нейронных сетей, М.: Горячая линия – Телеком, 2011. 408 с.
4. Г. Шилдт, C# 4.0. Полное руководство, Вильямс, 2013, 1056 с.
5. Уоссермен Ф. Нейрокомпьютерная техника. — М.: Мир, 1992. 127 с.

ПРИЛОЖЕНИЕ. КОД ПРОГРАММЫ

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace neuro

{

public partial class Form1 : Form

{

private int step;

private bool isFirstEra;

private double i1;

private double i2;

private double w04;

private double w05;

private double w14;

private double w15;

private double w24;

private double w25;

private double w36;

private double w46;

private double w56;

private double dw04;

private double dw05;

private double dw14;

private double dw15;

private double dw24;

private double dw25;

private double dw36;

private double dw46;

private double dw56;

private double o0;

private double o1;

private double o2;

private double o3;

private double o4;

private double o5;

private double o6;

private double net4;

private double net5;

private double net6;

private double gamma1;

private double gamma2;

private double th;

public Form1() {

InitializeComponent();

step = 0;

isFirstEra = true;

}

private void RenewParameters() {

i1 = Convert.ToDouble( numericUpDown\_i1.Value );

i2 = Convert.ToDouble( numericUpDown\_i2.Value );

w04 = Convert.ToDouble( numericUpDown\_w04.Value );

w05 = Convert.ToDouble( numericUpDown\_w05.Value );

w14 = Convert.ToDouble( numericUpDown\_w14.Value );

w15 = Convert.ToDouble( numericUpDown\_w15.Value );

w24 = Convert.ToDouble( numericUpDown\_w24.Value );

w25 = Convert.ToDouble( numericUpDown\_w25.Value );

w36 = Convert.ToDouble( numericUpDown\_w36.Value );

w46 = Convert.ToDouble( numericUpDown\_w46.Value );

w56 = Convert.ToDouble( numericUpDown\_w56.Value );

o0 = Convert.ToDouble( numericUpDown\_o0.Value );

o1 = Convert.ToDouble( numericUpDown\_o1.Value );

o2 = Convert.ToDouble( numericUpDown\_o2.Value );

o3 = Convert.ToDouble( numericUpDown\_o3.Value );

o4 = Convert.ToDouble( numericUpDown\_o4.Value );

o5 = Convert.ToDouble( numericUpDown\_o5.Value );

o6 = Convert.ToDouble( numericUpDown\_o6.Value );

net4 = Convert.ToDouble( numericUpDown\_net4.Value );

net5 = Convert.ToDouble( numericUpDown\_net5.Value );

net6 = Convert.ToDouble( numericUpDown\_net6.Value );

gamma1 = Convert.ToDouble( numericUpDown\_gamma1.Value );

gamma2 = Convert.ToDouble( numericUpDown\_gamma2.Value );

th = Convert.ToDouble( numericUpDown\_th.Value );

}

private void RenewNet4() {

RenewParameters();

numericUpDown\_net4.Value = Convert.ToDecimal(o0 \* w04 + o1 \* w14 + o2 \* w24);

}

private void RenewNet5() {

RenewParameters();

numericUpDown\_net5.Value = Convert.ToDecimal(o0 \* w05 + o1 \* w15 + o2 \* w25);

}

private void RenewNet6() {

RenewParameters();

numericUpDown\_net6.Value = Convert.ToDecimal(o3 \* w36 + o4 \* w46 + o5 \* w56);

}

private void RenewO0() {

numericUpDown\_o0.Value = 1;

}

private void RenewO1() {

numericUpDown\_o1.Value = numericUpDown\_i1.Value;

}

private void RenewO2() {

numericUpDown\_o2.Value = numericUpDown\_i2.Value;

}

private void RenewO3() {

numericUpDown\_o3.Value = 1;

}

private void RenewO4() {

numericUpDown\_o4.Value = Convert.ToDecimal(Math.Tanh(Convert.ToDouble(numericUpDown\_net4.Value)));

}

private void RenewO5() {

numericUpDown\_o5.Value = Convert.ToDecimal(Math.Tanh(Convert.ToDouble(numericUpDown\_net5.Value)));

}

private void RenewO6() {

numericUpDown\_o6.Value = Convert.ToDecimal(Math.Tanh(Convert.ToDouble(numericUpDown\_net6.Value)));

}

private void RenewOutput() {

RenewParameters();

numericUpDown\_output.Value = numericUpDown\_o6.Value;

if (Math.Abs(o6 - th) <= 0.001)

{

MessageBox.Show("ИНС обучена!");

button1.Enabled = false;

}

}

private void RenewW56() {

RenewParameters();

double delta6 = (th - o6) \* (o6 + 1) \* (1 - o6);

if (isFirstEra)

dw56 = gamma1 \* delta6 \* o5;

else

dw56 = gamma1 \* delta6 \* o5 + gamma2 \* dw56;

w56 = w56 + dw56;

numericUpDown\_w56.Value = Convert.ToDecimal(w56);

}

private void RenewW46() {

RenewParameters();

double delta6 = (th - o6) \* (o6 + 1) \* (1 - o6);

if (isFirstEra)

dw46 = gamma1 \* delta6 \* o4;

else

dw46 = gamma1 \* delta6 \* o4 + gamma2 \* dw46;

w46 = w46 + dw46;

numericUpDown\_w46.Value = Convert.ToDecimal(w46);

}

private void RenewW36() {

RenewParameters();

double delta6 = (th - o6) \* (o6 + 1) \* (1 - o6);

if (isFirstEra)

dw36 = gamma1 \* delta6 \* o3;

else

dw36 = gamma1 \* delta6 \* o3 + gamma2 \* dw36;

w36 = w36 + dw36;

numericUpDown\_w36.Value = Convert.ToDecimal(w36);

}

private void RenewW25() {

RenewParameters();

double delta6 = (th - o6) \* (o6 + 1) \* (1 - o6);

double delta5 = (o5 + 1) \* (1 - o5) \* delta6 \* w56;

double delta4 = (o4 + 1) \* (1 - o4) \* delta6 \* w46;

if (isFirstEra)

dw25 = gamma1 \* delta5 \* o2;

else

dw25 = gamma1 \* delta5 \* o2 + gamma2 \* dw25;

w25 = w25 + dw25;

numericUpDown\_w25.Value = Convert.ToDecimal(w25);

}

private void RenewW24() {

RenewParameters();

double delta6 = (th - o6) \* (o6 + 1) \* (1 - o6);

double delta5 = (o5 + 1) \* (1 - o5) \* delta6 \* w56;

double delta4 = (o4 + 1) \* (1 - o4) \* delta6 \* w46;

if (isFirstEra)

dw24 = gamma1 \* delta4 \* o2;

else

dw24 = gamma1 \* delta4 \* o2 + gamma2 \* dw24;

w24 = w24 + dw24;

numericUpDown\_w24.Value = Convert.ToDecimal(w24);

}

private void RenewW15() {

RenewParameters();

double delta6 = (th - o6) \* (o6 + 1) \* (1 - o6);

double delta5 = (o5 + 1) \* (1 - o5) \* delta6 \* w56;

double delta4 = (o4 + 1) \* (1 - o4) \* delta6 \* w46;

if (isFirstEra)

dw15 = gamma1 \* delta5 \* o1;

else

dw15 = gamma1 \* delta5 \* o1 + gamma2 \* dw15;

w15 = w15 + dw15;

numericUpDown\_w15.Value = Convert.ToDecimal(w15);

}

private void RenewW14() {

RenewParameters();

double delta6 = (th - o6) \* (o6 + 1) \* (1 - o6);

double delta5 = (o5 + 1) \* (1 - o5) \* delta6 \* w56;

double delta4 = (o4 + 1) \* (1 - o4) \* delta6 \* w46;

if (isFirstEra)

dw14 = gamma1 \* delta4 \* o1;

else

dw14 = gamma1 \* delta4 \* o1 + gamma2 \* dw14;

w14 = w14 + dw14;

numericUpDown\_w14.Value = Convert.ToDecimal(w14);

}

private void RenewW05() {

RenewParameters();

double delta6 = (th - o6) \* (o6 + 1) \* (1 - o6);

double delta5 = (o5 + 1) \* (1 - o5) \* delta6 \* w56;

double delta4 = (o4 + 1) \* (1 - o4) \* delta6 \* w46;

if (isFirstEra)

dw05 = gamma1 \* delta5 \* o0;

else

dw05 = gamma1 \* delta5 \* o0 + gamma2 \* dw05;

w05 = w05 + dw05;

numericUpDown\_w05.Value = Convert.ToDecimal(w05);

}

private void RenewW04() {

RenewParameters();

double delta6 = (th - o6) \* (o6 + 1) \* (1 - o6);

double delta5 = (o5 + 1) \* (1 - o5) \* delta6 \* w56;

double delta4 = (o4 + 1) \* (1 - o4) \* delta6 \* w46;

if (isFirstEra)

dw04 = gamma1 \* delta4 \* o0;

else

dw04 = gamma1 \* delta4 \* o0 + gamma2 \* dw04;

w04 = w04 + dw04;

numericUpDown\_w04.Value = Convert.ToDecimal(w04);

}

private void button1\_Click(object sender, EventArgs e) {

switch (step)

{

case 0:

button1.Text = "Далее";

numericUpDown\_i1.Enabled = false;

numericUpDown\_i2.Enabled = false;

numericUpDown\_w04.Enabled = false;

numericUpDown\_w05.Enabled = false;

numericUpDown\_w14.Enabled = false;

numericUpDown\_w15.Enabled = false;

numericUpDown\_w24.Enabled = false;

numericUpDown\_w25.Enabled = false;

numericUpDown\_w36.Enabled = false;

numericUpDown\_w46.Enabled = false;

numericUpDown\_w56.Enabled = false;

numericUpDown\_th.Enabled = false;

numericUpDown\_gamma1.Enabled = false;

numericUpDown\_gamma2.Enabled = false;

step = 1;

break;

case 1:

numericUpDown\_o0.BackColor = Color.Yellow;

numericUpDown\_o1.BackColor = Color.Yellow;

numericUpDown\_o2.BackColor = Color.Yellow;

RenewO0();

RenewO1();

RenewO2();

step = 2;

break;

case 2:

numericUpDown\_o0.BackColor = SystemColors.Control;

numericUpDown\_o1.BackColor = SystemColors.Control;

numericUpDown\_o2.BackColor = SystemColors.Control;

numericUpDown\_net4.BackColor = Color.Yellow;

numericUpDown\_net5.BackColor = Color.Yellow;

numericUpDown\_o3.BackColor = Color.Yellow;

numericUpDown\_o4.BackColor = Color.Yellow;

numericUpDown\_o5.BackColor = Color.Yellow;

RenewNet4();

RenewNet5();

RenewO3();

RenewO4();

RenewO5();

step = 3;

break;

case 3:

numericUpDown\_net4.BackColor = SystemColors.Control;

numericUpDown\_net5.BackColor = SystemColors.Control;

numericUpDown\_o3.BackColor = SystemColors.Control;

numericUpDown\_o4.BackColor = SystemColors.Control;

numericUpDown\_o5.BackColor = SystemColors.Control;

numericUpDown\_net6.BackColor = Color.Yellow;

numericUpDown\_o6.BackColor = Color.Yellow;

RenewNet6();

RenewO6();

step = 4;

break;

case 4:

numericUpDown\_net6.BackColor = SystemColors.Control;

numericUpDown\_o6.BackColor = SystemColors.Control;

numericUpDown\_output.BackColor = Color.Yellow;

RenewOutput();

step = 5;

break;

case 5:

numericUpDown\_net6.BackColor = Color.Yellow;

numericUpDown\_o6.BackColor = Color.Yellow;

numericUpDown\_output.BackColor = SystemColors.Control;

step = 6;

break;

case 6:

numericUpDown\_net6.BackColor = SystemColors.Control;

numericUpDown\_o6.BackColor = SystemColors.Control;

numericUpDown\_w56.BackColor = Color.Yellow;

numericUpDown\_w46.BackColor = Color.Yellow;

numericUpDown\_w36.BackColor = Color.Yellow;

RenewW56();

RenewW46();

RenewW36();

step = 7;

break;

case 7:

numericUpDown\_w56.BackColor = SystemColors.Control;

numericUpDown\_w46.BackColor = SystemColors.Control;

numericUpDown\_w36.BackColor = SystemColors.Control;

numericUpDown\_w25.BackColor = Color.Yellow;

numericUpDown\_w15.BackColor = Color.Yellow;

numericUpDown\_w05.BackColor = Color.Yellow;

numericUpDown\_w24.BackColor = Color.Yellow;

numericUpDown\_w14.BackColor = Color.Yellow;

numericUpDown\_w04.BackColor = Color.Yellow;

RenewW25();

RenewW24();

RenewW15();

RenewW14();

RenewW05();

RenewW04();

isFirstEra = false;

step = 8;

break;

case 8:

numericUpDown\_w25.BackColor = SystemColors.Control;

numericUpDown\_w15.BackColor = SystemColors.Control;

numericUpDown\_w05.BackColor = SystemColors.Control;

numericUpDown\_w24.BackColor = SystemColors.Control;

numericUpDown\_w14.BackColor = SystemColors.Control;

numericUpDown\_w04.BackColor = SystemColors.Control;

numericUpDown\_net4.BackColor = Color.Yellow;

numericUpDown\_net5.BackColor = Color.Yellow;

numericUpDown\_o3.BackColor = Color.Yellow;

numericUpDown\_o4.BackColor = Color.Yellow;

numericUpDown\_o5.BackColor = Color.Yellow;

RenewNet4();

RenewNet5();

RenewO3();

RenewO4();

RenewO5();

step = 3;

break;

}

}

}

}