Министерство науки и высшего образования Российской Федерации
государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Проект «Нейронная сеть для регрессионного моделирования»

Руководитель СКБ Е.П. Жарикова

Подпись/дата

Ответственный исполнитель В.А. Машевский

Подпись/дата

2021

Карточка проекта

Название	Проект «Нейронная сеть для регрессионного моделирования»
Тип проекта	Инициативный (инициативный, по заказу, в рамках конкурса, учебная работа, другое)
Исполнители	В.А. Машевский ответственный исполнитель
Срок реализации	06.2021 Месяц, год

Использованные программные средства

Наименование	**Версия**
Pytorch	**1.9.0**
Windows	**10**

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

З А Д А Н И Е
на разработку

Выдано студентам:
Машевскому владимиру Антоновичу
Название проекта:
Нейронная сеть для регрессионного моделирования
Назначение: Программное обеспечение предназначено для решения задачи регрессионного моделирования.
Применение: Программное обеспечение может быть применено как аналог численно-аналитической модели деформирования элемента плоской оболочечной конструкции в окрестности концентрации напряжений.

План работ:

Этап	Дата начала	Дата окончания
Формирование требований к программному обеспечению	01.06.2021	03.06.2021
Разработка структуры программного обеспечения	03.06.2021	07.06.2021
Разработка и утверждение технического задания	01.06.2021	03.06.2021
Программная реализация	07.06.2021	19.06.2021
Тестирование и отладка системы	19.06.2021	23.06.2021
Подготовка документации	23.06.2021	25.06.2021
Опытная эксплуатация	25.06.2021	27.06.2021

Руководитель СКБ Е.П. Жарикова

Подпись/дата

Министерство науки и высшего образования Российской Федерации
государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Руководство администратора
Проект «Нейронная сеть для регрессионного моделирования»

Руководитель СКБ Е.П. Жарикова

Подпись/дата

Ответственный исполнитель В.А. Машевский

Подпись/дата

2021

Содержание

1 Общие положения

Настоящий документ представляет собой руководство администратора программного обеспечения «Нейронная сеть для регрессионного моделирования» далее ПО. Руководство определяет порядок установки, настройки и администрирования ПО. Перед установкой и эксплуатацией системы рекомендуется внимательно ознакомиться с настоящим руководством. Документ подготовлен в соответствии с РД 50-34.698-90 - в части структуры и содержания документов, и в соответствии с ГОСТ 34.201-89 - в части наименования и обозначения документов.

1.1 Наименование программы

Наименование программного продукта: программное обеспечение для решения задачи регрессионного моделирования.

1.2 Наименование документов, на основании которых ведется проектирование системы

Создание ПО осуществляется на основании требований и положений следующих документов: задание.

1.3 Перечень организаций, участвующих в разработке системы

Разработчики: студент Комсомольского-на-Амуре государственного технического университета В.А. Машевский.

Заказчик: Комсомольский-на-Амуре государственный университет, студенческое конструкторское бюро «Интеллектуальные технологии».

1.4 Сведения об использованных при проектировании нормативно-технических документах

При проектировании использованы следующие нормативно-технические документы:

- ГОСТ 19.101-77 – виды программ и программных документов.

- ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом.

- ГОСТ 19.201-78 Техническое задание, требования к содержанию и оформлению;

- ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению.

- ГОСТ 2.004-88. Единая система конструкторской документации. Общие требования к выполнению конструкторских технологических документов на печатающих и графических устройствах вывода ЭВМ.

2 Назначение и принцип действия

2.1 Назначение изделия

Программное обеспечение для регрессионного моделирования

2.2 Области использования изделия

Программное обеспечение может быть применено как аналог численно-аналитической модели деформирования элемента плоской оболочечной конструкции в окрестности концентрации напряжений.

2.3 Принцип действия

Программное обеспечение решает задачу регрессии при помощи нейронной сети.

3. Описание программного обеспечения

3.1 Описание работы программного обеспечения

Результат программы - графики зависимости первого и второго инварианта тензора деформации Альманси от относительного удлинения образца при непрерывном поле скоростей деформации (Рисунок 1)

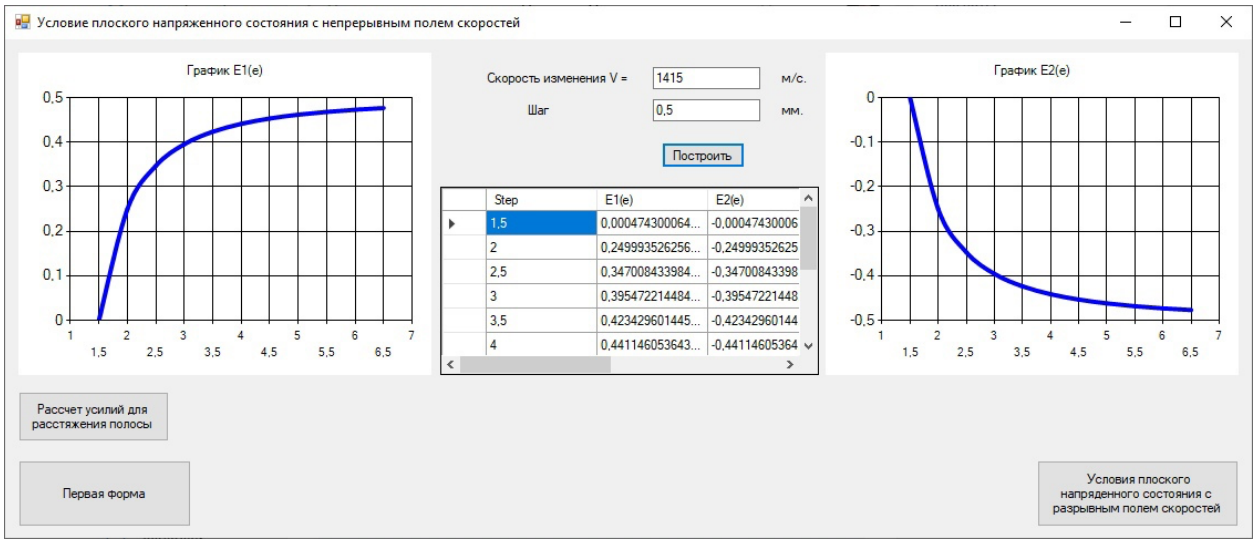

Рисунок 1 – Результат работы программы

3.2 Входные данные

Скорость изменения, шаг.

3.3 Выходные данные

Грифик зависимости первого и второго инварианта тензора деформации Альманси от относительного удлинения образца при непрерывном поле скоростей деформации.

4. Текст программы

Реализация представлена в листинге 1.

Листинг 1 – «Программа запуска системы»

Form1

```
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 _3d
{
   /// <summary>
   /// класс основной формы
   /// </summary>
   public partial class Form_main : Form
   {

      # region переменные

      // угол
      double angel_OXY;

      // угол
      double angel_res_OXY;
```

```
// точка 0
Point Point_0 = new Point(0, 0);

// фигура
List<Point3D> figure_3D = new List<Point3D>();

// pen для проекции figure_3D
Pen pen_figure_3D = new Pen(Color.Red);

// для временного хранения при поворотах
int tmp_XX;
int tmp_YY;

# endregion

/// <summary>
/// конструктор формы
/// </summary>
public Form_main()
{
    InitializeComponent();

    // Двойная буф-я
    typeof(Control).GetProperty("DoubleBuffered", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Instance | System.Reflection.BindingFlags.SetProperty).SetValue(pictureBox_main, true, null);

    // зададим точку отсчета по середине
    Point_0.X = pictureBox_main.Width / 2;
```

```
            Point_0.Y = pictureBox_main.Height / 2;

            // установим углы
            angel_OXY = 1.0;
            angel_res_OXY = 1.0;

            //
            trackBar_OX.Value = 100;
            trackBar_res_OXY.Value = 100;

            InitDataForComboBox();
        }

        # region расчеты и отрисовка

        // проекция 2D на 2D
        private Point convert_3D_in_2D_Point(Point3D val)
        {
            // проицируем
            double res_x = -val._z * System.Math.Sin(angel_OXY) + val._x * Sys-
tem.Math.Cos(angel_OXY) + Point_0.X;
            double res_y = -(val._z * System.Math.Cos(angel_OXY) + val._x * Sys-
tem.Math.Sin(angel_OXY)) * System.Math.Sin(angel_res_OXY) + val._y * Sys-
tem.Math.Cos(angel_res_OXY) + Point_0.Y ;

            return new Point((int)(res_x), (int)(res_y));
        }

        # endregion
```

```
    # region фигуры

    // рисуем
    void draw(List<Point3D> val)
    {
        // проверка наличия фигуры 3d
        if (figure_3D.Count <= 0)
            return;

        // создадим bitmap и Graphics
        Bitmap bmp = new Bitmap(pictureBox_main.Width, pictureBox_main.Height);
        Graphics grf = Graphics.FromImage(bmp);



        // пербираем
        for (int i = 0; i < val.Count - 1; i++)
        {
            grf.DrawLine(pen_figure_3D, convert_3D_in_2D_Point(val[i]), convert_3D_in_2D_Point(val[i + 1]));
        }

        // выводим
        pictureBox_main.Image = bmp;
        pictureBox_main.Refresh();
        GC.Collect();

    }

    // куб
```

```
private void cube()
{
    // очистим если есть
    if (figure_3D != null)
        figure_3D.Clear();

    // заполним
    figure_3D.Add(new Point3D(0, 0, 0));
    figure_3D.Add(new Point3D(300, 0, 0));
    figure_3D.Add(new Point3D(300, 0, 5));
    figure_3D.Add(new Point3D(0, 0, 5));
    figure_3D.Add(new Point3D(0, 400, 5));
    figure_3D.Add(new Point3D(0, 400, 0));
    figure_3D.Add(new Point3D(0, 0, 5));
    figure_3D.Add(new Point3D(0, 0, 0));
    figure_3D.Add(new Point3D(0, 400, 0));
    figure_3D.Add(new Point3D(300, 400, 0));
    figure_3D.Add(new Point3D(300, 400, 5));
    figure_3D.Add(new Point3D(300, 0, 5));
    figure_3D.Add(new Point3D(300, 0, 0));
    figure_3D.Add(new Point3D(300, 400, 0));
    figure_3D.Add(new Point3D(0, 400, 0));
    figure_3D.Add(new Point3D(0, 400, 5));
    figure_3D.Add(new Point3D(300, 400, 5));
    figure_3D.Add(new Point3D(300, 400, 5));
}

# endregion

# region обработчики
```

```
        // куб
        private void button_cub_Click(object sender, EventArgs e)
        {
            cube();
            draw(figure_3D);
        }

        // для DOWNmous
        private void pictureBox_main_MouseDown(object sender, MouseEventArgs
e)
        {
            // зададим точку отсчета
            if (e.Button == System.Windows.Forms.MouseButtons.Left)
            {
                Point_0.X = e.X;
                Point_0.Y = e.Y;
            }
        }

        private void pictureBox_main_MouseMove(object sender, MouseEventArgs
e)
        {
            if (e.Button == System.Windows.Forms.MouseButtons.Right)
            {
                # region горизонтальный поворт
                // вправо
                if (e.X > tmp_XX)
                {
```

```
            if (trackBar_OX.Value <= trackBar_OX.Maximum && track-
Bar_OX.Value > 0)
            {
               trackBar_OX.Value -= 1;
               trackBar1_Scroll(this, EventArgs.Empty);
            }
            else
            {
               if (trackBar_OX.Value == 0)
                  trackBar_OX.Value = trackBar_OX.Maximum;
            }
         }

         // влево
         if (e.X < tmp_XX)
         {
            if (trackBar_OX.Value < trackBar_OX.Maximum && track-
Bar_OX.Value >= 0)
            {
               trackBar_OX.Value += 1;
               trackBar1_Scroll(this, EventArgs.Empty);

               // переход через MAX
               if (trackBar_OX.Value >= trackBar_OX.Maximum)
                  trackBar_OX.Value = 0;
            }
         }

         # endregion
```

```
            # region вертикальный поворт


            // вниз
            if (e.Y > tmp_YY)
            {
               if (trackBar_res_OXY.Value < trackBar_res_OXY.Maximum &&
trackBar_res_OXY.Value >= 0)
               {
                  trackBar_res_OXY.Value += 1;
                  trackBar_res_OXY_Scroll(this, EventArgs.Empty);
               }
               else
               {
                  // переход через MAX
                   if (trackBar_res_OXY.Value >= trackBar_OX.Maximum)
                      trackBar_res_OXY.Value = 0;
               }
            }


            // вверх
            if (e.Y < tmp_YY)
            {
               if (trackBar_res_OXY.Value <= trackBar_res_OXY.Maximum &&
trackBar_res_OXY.Value > 0)
               {
                  trackBar_res_OXY.Value -= 1;
                  trackBar_res_OXY_Scroll(this, EventArgs.Empty);
               }
               else
               {
```

```
                // переход через 0
                if (trackBar_res_OXY.Value <= 0)
                    trackBar_res_OXY.Value = trackBar_res_OXY.Maximum;
            }
        }

        # endregion

        // временно для хранения
        tmp_XX = e.X;
        tmp_YY = e.Y;

        return;
    }

    // пермещаем точку отсчета 0
    if (e.Button == System.Windows.Forms.MouseButtons.Left)
    {
        Point_0.X = e.X;
        Point_0.Y = e.Y;
        draw(figure_3D);
    }
}

// поворот
private void trackBar1_Scroll(object sender, EventArgs e)
{
    angel_OXY = (double)(trackBar_OX.Value) / 100;
    draw(figure_3D);
}
```

```
// поворот
private void trackBar_res_OXY_Scroll(object sender, EventArgs e)
{
   angel_res_OXY = (double)(trackBar_res_OXY.Value) / 100;
   draw(figure_3D);
}

// изменение размеров формы
private void Form_main_Resize(object sender, EventArgs e)
{
   draw(figure_3D);
}

#endregion

private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
   if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))
   {
      return;
   }
   if(Char.IsControl(e.KeyChar))
   {
      button1.Focus();
      return;
   }
   e.Handled = true;
}
public void button2_Click(object sender, EventArgs e)
```

```
    {

    }

    private void button1_Click(object sender, EventArgs e)
    {
        Form2 f = new Form2();
        f.textBox1.Text = this.textBox1.Text;

        Keeper.value1 = textBox2.Text; //ширина
        Keeper.value2 = textBox3.Text; //толщина
        Keeper.value3 = label5.Text;   //коэффициент к

        f.Show();
        Hide();
    }

    int a;
    public void InitDataForComboBox()
    {
        var dataSource1 = new List<ClassNumber>();
        dataSource1.Add(new ClassNumber() { IdNumber = 50, NameNumber =
"Алюминий" });
        dataSource1.Add(new ClassNumber() { IdNumber = 70, NameNumber =
"Медь" });
        dataSource1.Add(new ClassNumber() { IdNumber = 7, NameNumber =
"Свинец" });

        comboBox1.DataSource = dataSource1;
        comboBox1.DisplayMember = "NameNumber";
```

```
            comboBox1.ValueMember = "IdNumber";

            comboBox1.SelectedIndexChanged += new System.EventHandler(com-
boBox1_SelectedIndexChanged);
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs
e)
        {
            a = Convert.ToInt32(comboBox1.SelectedValue);
            label5.Text = comboBox1.SelectedValue.ToString();
        }

        private void label4_Click(object sender, EventArgs e)
        {

        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```