ДОДАТОК А

Програмний код

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Reflection;
using ExcelObj = Microsoft.Office.Interop.Excel;
using System.IO;
using System.Data.SqlClient;
namespace Донец
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        class ClassNS
        {

            public static int newnna = Data.newnna;
            public static int nna = Data.nna;
            public static int nvih = Data.nvih;
            public static int nvh = Data.nvh;
            public static int nres = Data.nres;
            public double[][] B = new double[ne - nv + 1][];

            public double[] q = new double[ne - nv + 1];
            public double[] q1 = new double[Data.ntab1];
            public double[] h = new double[Data.ntab1];
            public double[] dhdq = new double[Data.ntab1];
            public double[] l = new double[nna * 5 + 2 * (nna - 1) + Data.nvih +
Data.nvh];
            public double[] d = new double[nna * 5 + 2 * (nna - 1) + Data.nvih +
Data.nvh];
            public double[] dh = new double[nna * 5 + 2 * (nna - 1) + Data.nvih +
Data.nvh];
            public double[,] NA = new double[nna, 10];
            public double[,] RZ = new double[nna, 9];
            public double[,] Pass = new double[nna * 4 - 1, 7];
            public double[,] vihod = new double[Data.nvih, 5];
            public double[,] vhod = new double[Data.nvh, 5];
            public double v(int i, double q11)
            {
                double v1;
                //м/с
                v1 = 4 * q11 / (3.14 * d[i] * d[i]);
                return v1;
            }
            public int susduga(int[,] mass, int duganach, int dugakon)
            {
                //возвращает 1/0, если в массиве в столбцах 0,1 есть/нет дуга
duganach-dugakon
                int susduga = 0;
                for (int i = 0; i < mass.GetLength(0); i++)
```

```
        1; break; }
                    }
                    return susduga;
                }

                public int ind2(int[,] mass, int n, int duga)
                {
                    //возвращает индекс єлемента duga в n-ом столбце массива
                    int ind = 0;
                    for (int i = 0; i < mass.GetLength(0); i++)
                    {
                        if (mass[i, n] == duga) { ind = i; break; }
                    }
                    return ind;
                }
                public int indduga(int[,] mass, int uz1, int uz2)
                {
                    //возвращает индекс дуги uz1-uz2 uz2-uz1
                    int indduga = 0;
                    if (susduga(mass, uz1, uz2) == 1)
                    {
                        for (int i = 0; i < mass.GetLength(0); i++)
                        {
                            if (mass[i, 0] == uz1 & mass[i, 1] == uz2) { indduga = i;
break;
                            }
                    }
                    if (susduga(mass, uz2, uz1) == 1)
                    {
                        for (int i = 0; i < mass.GetLength(0); i++)
                        {
                            if (mass[i, 0] == uz2 & mass[i, 1] == uz1) { indduga = i;
break;
                            }
                    }
                    return indduga;
                }
                public double S(int i)
                {
                    double S1;
                      S1 = (0.00579 * l[i] / (Math.Pow(d[i], 5)));
                    return S1;
                }
                public double dS(int i)
                {
                    double dS1;
                    { dS1 = 0; }
                    return dS1;
                }
                public double[] qqq(double[] q33)
                {
          w1:               for (int i = 0; i < Data.nu2; i++)
                    {
                        q[i] = q33[i];
                  }
                    for (int i = Data.nu2 + Data.nvih; i < Data.nu2 + Data.nvih +
Data.newnvhod - 1; i++)
                        q[i] = q33[i - Data.nvih];
                    }
                    //B[nna+nvih-1, ne-nvih]
                    for (int j = 0; j < Data.ntab1; j++)
                    {
                        double sum = 0;
                        for (int i = 0; i < B.GetLength(0); i++)
                        {
```

```
                            sum = sum + B[i][j] * q[i];
                        }
                        q1[j] = sum;

                    }
                    for (int i = 0; i < Data.newnna; i++)
                    {
                        h[Convert.ToInt16(Data.NA[i,   15])]   =   -(Data.NA[i,   6]   *
Math.Pow(Data.NA[i, 5] / Data.NA[i, 4], 2) + Data.NA[i, 7] * Data.NA[i, 5] /
                                Data.NA[i,   4]   *   q1[Convert.ToInt16(Data.NA[i,   15])]   +
Data.NA[i, 8] * q1[Convert.ToInt16(Data.NA[i, 15])] * q1[Convert.ToInt16(Data.NA[i, 15])]) +
Data.RZ[i, 7] * q1[Convert.ToInt16(Data.NA[i, 15])] / Math.Pow(Data.RZ[i, 8], 2);
                        dhdq[Convert.ToInt16(Data.NA[i,   15])]   =   -(Data.NA[i,   7]   *
Data.NA[i, 5] / Data.NA[i, 4] + 2 * Data.NA[i, 8] * q1[Convert.ToInt16(Data.NA[i, 15])]) +
Data.RZ[i, 7] / Math.Pow(Data.RZ[i, 8], 2);
                            }
                    for (int i = 0; i < Data.Pass.GetLength(0); i++)
                    {
                        h[Convert.ToInt16(Data.Pass[i,               9])]               =
S(Convert.ToInt16(Data.Pass[i,    9]))    *    q1[Convert.ToInt16(Data.Pass[i,    9])]    *
Math.Abs(q1[Convert.ToInt16(Data.Pass[i, 9])]);
                        dhdq[Convert.ToInt16(Data.Pass[i, 9])] =/* dS(ndugi, q1[ndugi]) *
q1[ndugi]    *    Math.Abs(q1[ndugi])    +*/    2    *    S(Convert.ToInt16(Data.Pass[i,    9]))
*Math.Sign(q1[Convert.ToInt16(Data.Pass[i,                    9])])                    *
/*Math.Abs*/(q1[Convert.ToInt16(Data.Pass[i, 9])]);
                    }
                    for (int i = 0; i < Data.vhod.GetLength(0); i++)
                    {
                        h[Convert.ToInt16(Data.vhod[i,               8])]               =
(Math.Abs(q1[Convert.ToInt16(Data.vhod[i,  8])])  *  q1[Convert.ToInt16(Data.vhod[i,  8])]  *
S(Convert.ToInt16(Data.vhod[i, 8])));
                        dhdq[Convert.ToInt16(Data.vhod[i,              8])]              =
(q1[Convert.ToInt16(Data.vhod[i,  8])]  *  2  */*  Math.Sign(q1[Convert.ToInt16(Data.vhod[i,
8])]) */ S(Convert.ToInt16(Data.vhod[i, 8])));
                    }
                   return q1;
                public int elvmass(int[] mass, int el)
                {
                    //проверяет есть ли заданный элемент в массиве
                    int elvmass = 0;
                    for (int i = 0; i < mass.GetLength(0); i++)
                    {
                        if (el == mass[i]) elvmass = 1;
                    }
                    return elvmass;
                }
                public double[] f(double[] q33)
                {
                    double[] ff = new double[Data.nu2 + Data.newnvhod - 1];

                    qqq(q33);
                    int i1 = 0;
                    for (int i = 0; i < Data.nu2; i++)
                    {
                        double sum = 0;
                        for (int j = 0; j < Data.ntab1; j++)
                        {
                            sum = sum + B[i][j] * h[j];
                        }
                        ff[i1] = sum; i1 = i1 + 1;

                    }
                    for  (int  i  =  Data.nu2  +  Data.nvih;  i  <  Data.nu2  +  Data.nvih  +
Data.newnvhod - 1; i++)
                    {
```

```
                        double sum = 0;
                        for (int j = 0; j < Data.ntab1; j++)
                        {
                            sum = sum + B[i][j] *( h[j]+dh[j]);
                        }
                        ff[i1] = Data.hvh[0]  - Data.hvh[i - (Data.nu2 + Data.nvih) + 1]
+ sum; i1 = i1 + 1;
                    }
                    return ff;

                }
                public double[][] df(double[] q33)
                {
                    double[][] dff = MatrixCreate(Data.nu2 + Data.newnvhod - 1, Data.nu2
+ Data.newnvhod - 1);
                    qqq(q33);
                    int k1 = 0;
                    for (int k = 0; k < Data.nu2 + Data.nvih + Data.newnvhod - 1; k++)
                    {
                    kk1: if (k >= Data.nu2 & k < Data.nu2 + Data.nvih)
                        { k = k + 1; if (k == Data.nu2 + Data.nvih + Data.newnvhod - 1) {
break; } goto kk1; }
                        int i1 = 0;
                        for (int i = 0; i < Data.nu2 + Data.nvih + Data.newnvhod - 1;
i++)
                        {
                        kk2: if (i >= Data.nu2 & i < Data.nu2 + Data.nvih)
                            { i = i + 1; if (i == Data.nu2 + Data.nvih + Data.newnvhod -
1) { break; } goto kk2; }
                            double sum = 0;
                            for (int j = 0; j < Data.ntab1; j++)
                            {

                                sum = sum + B[k][j] * B[i][j] * dhdq[j];
                            }
                            dff[k1][i1] = sum; i1 = i1 + 1;
                        }
                        k1 = k1 + 1;
                    }
                    return dff;
                /// <returns></returns>
                public double[][] MatrixCreate(int rows, int cols)
                {
                    // Создаем матрицу, полностью инициализированную
                    // значениями 0.0. Проверка входных параметров опущена.
                    double[][] result = new double[rows][];
                    for (int i = 0; i < rows; ++i)
                        result[i] = new double[cols]; // автоинициализация в 0.0
                    return result;
                }
                public string MatrixAsString(double[][] matrix)
                {
                    string s = "";
                    for (int i = 0; i < matrix.Length; ++i)
                    {
                        for (int j = 0; j < matrix[i].Length; ++j)
                            s += matrix[i][j].ToString("F0").PadLeft(8) + " ";
                        s += Environment.NewLine;
                    }
                    return s;
                }
                public double[][] MatrixProduct(double[][] matrixA,
            double[][] matrixB)
                {
                    int aRows = matrixA.Length; int aCols = matrixA[0].Length;
```

```
            int bRows = matrixB.Length; int bCols = matrixB[0].Length;
            if (aCols != bRows)
                throw new Exception("Non-conformable matrices in MatrixProduct");
            double[][] result = MatrixCreate(aRows, bCols);
            for (int i = 0; i < aRows; ++i) // каждая строка A
                for (int j = 0; j < bCols; ++j) // каждый столбец B
                    for (int k = 0; k < aCols; ++k)
                        result[i][j] += matrixA[i][k] * matrixB[k][j];
            return result;
        }
        public double[][] MatrixDecompose(double[][] matrix,
    out int[] perm, out int toggle)
        {
            // Разложение LUP Дулитла. Предполагается,
            // что матрица квадратная.
            int n = matrix.Length; // для удобства
            double[][] result = MatrixDuplicate(matrix);
            perm = new int[n];
            for (int i = 0; i < n; ++i) { perm[i] = i; }
            toggle = 1;
            for (int j = 0; j < n - 1; ++j) // каждый столбец
            {
                double colMax = Math.Abs(result[j][j]); // Наибольшее значение в
столбце j
                int pRow = j;
                for (int i = j + 1; i < n; ++i)
                {
                    if (result[i][j] > colMax)
                    {
                        colMax = result[i][j];
                        pRow = i;
                    }
                }
                if (pRow != j) // перестановка строк
                {
                    double[] rowPtr = result[pRow];
                    result[pRow] = result[j];
                    result[j] = rowPtr;
                    int tmp = perm[pRow]; // Меняем информацию о перестановке
                    perm[pRow] = perm[j];
                    perm[j] = tmp;
                    toggle = -toggle; // переключатель перестановки строк
                }
                if (Math.Abs(result[j][j]) < 1.0E-20)
                    return null;
                for (int i = j + 1; i < n; ++i)
                {
                    result[i][j] /= result[j][j];
                    for (int k = j + 1; k < n; ++k)
                        result[i][k] -= result[i][j] * result[j][k];
                }
            } // основной цикл по столбцу j
            return result;
        }
        public double MatrixDeterminant(double[][] matrix)
        {
            int[] perm;
            int toggle;
            double[][] lum = MatrixDecompose(matrix, out perm, out toggle);
            if (lum == null)
                throw new Exception("Unable to compute MatrixDeterminant");
            double result = toggle;
            for (int i = 0; i < lum.Length; ++i)
                result *= lum[i][i];
            return result;
```

```
        }
        public double[] HelperSolve(double[][] luMatrix,
double[] b)
        {
            // Решаем luMatrix * x = b
            int n = luMatrix.Length;
            double[] x = new double[n];
            b.CopyTo(x, 0);
            for (int i = 1; i < n; ++i)
            {
                double sum = x[i];
                for (int j = 0; j < i; ++j)
                    sum -= luMatrix[i][j] * x[j];
                x[i] = sum;
            }
            x[n - 1] /= luMatrix[n - 1][n - 1];
            for (int i = n - 2; i >= 0; --i)
            {
                double sum = x[i];
                for (int j = i + 1; j < n; ++j)
                    sum -= luMatrix[i][j] * x[j];
                x[i] = sum / luMatrix[i][i];
            }
            return x;
        }
        public double[][] MatrixInverse(double[][] matrix)
        {
            int n = matrix.Length;
            double[][] result = MatrixDuplicate(matrix);
            int[] perm;
            int toggle;
            double[][] lum = MatrixDecompose(matrix, out perm, out toggle);
            if (lum == null)
                throw new Exception("Unable to compute inverse");
            double[] b = new double[n];
            for (int i = 0; i < n; ++i)
            {
                for (int j = 0; j < n; ++j)
                {
                    if (i == perm[j])
                        b[j] = 1.0;
                    else
                        b[j] = 0.0;
                }
                double[] x = HelperSolve(lum, b);
                for (int j = 0; j < n; ++j)
                    result[j][i] = x[j];
            }
            return result;
        }
        public double[][] MatrixDuplicate(double[][] matrix)
        {
            // Предполагается, что матрица не нулевая
            double[][] result = MatrixCreate(matrix.Length, matrix[0].Length);
            for (int i = 0; i < matrix.Length; ++i) // Копирование значений
                for (int j = 0; j < matrix[i].Length; ++j)
                    result[i][j] = matrix[i][j];
            return result;
        }
    }
    public int[,] tree(int[,] tab1)
    {
        //возвращает дерево графа
        int[,] tree = new int[Data.newnv-1, 3];
        int n = 0, v = 0;
```

```
                for (int i = 0; i < Data.vhod.GetLength(0); i++)
                {
                    if                (Data.vhod[i,                4]                !=
10&susduga(Data.tab1,0,Convert.ToInt16(Data.vhod[i,0]))==1)
                    { tree[n, 0] = 0; tree[n, 1] = Convert.ToInt16(Data.vhod[i, 0]); n =
n + 1; break; }
                }

                for (int i = 0; i < tree.GetLength(0); i++)
                {
                    v = tree[i, 1];
                    for (int j = 0; j < Data.ntab1; j++)
                    {
                        if (tab1[j, 0] == v & susuzel2(tree, tab1[j, 1]) == 0)
                        {  tree[n,  0]  =  v;  tree[n,  1]  =  tab1[j,  1];  n  =  n  +  1;if
(n==tree.GetLength(0)) goto c13; }
                        if (tab1[j, 1] == v & susuzel2(tree, tab1[j, 0]) == 0)
                        { tree[n, 0] = v; tree[n, 1] = tab1[j, 0]; n = n + 1; if (n ==
tree.GetLength(0)) goto c13; }
                    }

                }
                //нумерует ветви дерева в Data.tab1
                c13:
                for (int i = 0; i < Data.ntab1; i++)
                {
                    Data.tab1[i, 2] = indduga(tree, Data.tab1[i, 0], Data.tab1[i, 1]);
                }
                //нумерует реальные хорды в Data.tab1
                int nu2 = 0;
                //int t = tree.GetLength(0);
                for (int i = 0; i < Data.ntab1; i++)
                {
                    if (Data.tab1[i, 0] != 0 & Data.tab1[i, 1] != 0 & Data.tab1[i, 2] ==
0)
                    { Data.tab1[i, 2] = n; n = n + 1; nu2 = nu2 + 1; }
                }
                Data.nu2 = nu2;
                //нумерует фиктивные хорды
                for (int i = 0; i < Data.ntab1; i++)
                {
                    if  (Data.tab1[i,  1]  ==  0  &  susduga(tree,  Data.tab1[i,  0],
Data.tab1[i, 1]) == 0)
                    { Data.tab1[i, 2] = n; n = n + 1; }
                }
                //нумерует фиктивные хорды от РЧВ
                int ksi2 = 0;
                for (int i = 0; i < Data.ntab1; i++)
                {
                    if  (Data.tab1[i,  0]  ==  0  &  susduga(tree,  Data.tab1[i,  0],
Data.tab1[i, 1]) == 0)
                    { Data.tab1[i, 2] = n; n = n + 1; ksi2 = ksi2 + 1; }
                }
                Data.ksi2 = ksi2;
                //нумерует узлы в Data.tab2
                Data.tab2 = sorttab2(Data.tab2, 0);
                Data.tab1 = sorttab1(Data.tab1, 2);


                return tree;
            }
                // расчёт м Ньютона
                for (int k = 0; k < M; k++)
                {
```

```
                    double[] func = new double[Data.nu2 + Data.newnvhod - 1];
                    func = p.f(q33);
                    //дача3

                    //
                    double[][] dfunc = p.MatrixCreate(Data.nu2 + Data.newnvhod - 1,
Data.nu2 + Data.newnvhod - 1);
                    dfunc = p.df(q33);
                    double[][] func1 = p.MatrixCreate(Data.nu2 + Data.newnvhod - 1,
Data.nu2 + Data.newnvhod - 1);
                    for (int i = 0; i < Data.nu2 + Data.newnvhod - 1; i++)
                    {
                        func1[i][0] = func[i];

                    }
                    //
                    int kkk = 0;
                    for (int i6 = 0; i6 < Data.nu2 + Data.newnvhod - 1; i6++)
                    {
                        if ((Convert.ToBoolean(Math.Abs(func[i6]) < ee)) == true)
                            kkk = kkk + 1;
                    }
                    if (kkk == Data.nu2 + Data.newnvhod - 1) goto vihod;



                    else
                    {

                        double[][] Wf = p.MatrixCreate(Data.nu2 + Data.newnvhod - 1,
Data.nu2 + Data.newnvhod - 1);

                        Wf = p.MatrixProduct(p.MatrixInverse(dfunc), func1);
                        //q3336 = q33;
                        for (int i = 0; i < Data.nu2 + Data.newnvhod - 1; i++)
                        {
                            q3336[i] = q3336[i] - Wf[i][0];
                            //   label6.Text = label6.Text + "\r\n" + Convert.ToString(
"ит " + (k + 1) + "  q3336[" + (i + 33) + "]=" + Math.Round(q3336[i], 3));

                        }

                        q33 = q3336;

                    }

                }

            vihod:
                {

                Data.h = p.h; Data.qq1 = p.q1;
                //запись q1 в DataGridView
                for (int i = 0; i < Data.vhod.GetLength(0); i++)
                {
                    dataGridView1[2,   ind(Data.vhod0,   1,   Convert.ToInt16(Data.vhod[i,
1]))].Value = p.q1[Convert.ToInt16(Data.vhod[i, 8])];

                }


                for (int i = 0; i < Data.Pass.GetLength(0); i++)
                {
```

```
                    dataGridView4[5,   ind(Data.Pass0,   8,   Convert.ToInt16(Data.Pass[i,
8]))].Value = p.q1[Convert.ToInt16(Data.Pass[i, 9])];
                }



                    //ветви дерева
                    for (int i = 0; i < Data.newnv - 1; i++)
                    {
                        ucPass01[eltab(Data.tab33,      0,      i,      1)].Text      =
ucPass01[eltab(Data.tab33, 0, i, 1)].Text + "\r\n" + "qд" + i + "=" + Math.Round(p.q1[i], 4)
+ "\r\n" + "h=" + Math.Round(p.h[i], 4) + "\r\n" /*+ "dh=" + p.dh[i]*/;

                        ////для картинки

                    }
                    //фикт хорді от активніх источников
                    for (int i = 0; i < Data.ksi2; i++)
                    {
                        ucPass01[eltab(Data.tab33,  0,  i  +  Data.ntab1  -  Data.ksi2,
1)].Text = ucPass01[eltab(Data.tab33, 0, i + Data.ntab1 - Data.ksi2, 1)].Text + "\r\n" +
"qa" + (i + Data.ntab1 - Data.ksi2) + "=" + Math.Round(p.q1[i + Data.ntab1 - Data.ksi2], 4)
+
                "\r\n" + "h=" + Math.Round(p.h[i + Data.ntab1 - Data.ksi2], 4) + "\r\n" +
"Нвх="  +  Math.Round(Data.hvh[i  +  1],  2)  +  "\r\n"  /*+  "dh="  +  eltab2(Data.vhod,  7,
eltab(Data.tab33, 0, i + Data.ntab1 - Data.ksi2, 1), 1)*/;

                        //для картинки



                    }
                    //реаль хорды
                    for (int i = 0; i < Data.nu2; i++)
                    {
                        ucPass01[eltab(Data.tab33,  0,  Data.newnv  -  1  +  i,  1)].Text  =
ucPass01[eltab(Data.tab33,  0,  Data.newnv  -  1  +  i,  1)].Text  +  "\r\n"  +  "qx="  +
Math.Round(p.q1[Data.newnv - 1 + i], 4) + "\r\n" + "h=" + Math.Round(p.h[Data.newnv - 1 +
i], 4) + "\r\n"/* + "dh=" + p.dh[Data.newnv - 1 + i]*/;

                        ////для картинки

                    }
                    //выходы
                    for (int i = 0; i < Data.nvih; i++)
                    {
                        ucPass01[eltab(Data.tab33,  0,  Data.nu2  +  Data.newnv  -  1  +  i,
1)].Text = ucPass01[eltab(Data.tab33, 0, Data.nu2 + Data.newnv - 1 + i, 1)].Text + "\r\n" +
"qв=" + Math.Round(p.q1[Data.nu2 + Data.newnv - 1 + i], 4);

                    }

                      ucPass01[      Convert.ToInt16(eltab2(Data.vhod,1,0,8))].Text      =
ucPass01[Convert.ToInt16(eltab2(Data.vhod,1,0,8))].Text          +          "\r\n"          +
"Нвх0="+eltab2(Data.vhod,3,0,8);


                    Data.q1 = p.q1;
                    //мощность НА
                    for (int k = 0; k < Data.NA.GetLength(0); k++)
                    {
                        S     =     S     +     Mosh(Convert.ToInt16(Data.NA[k,     1]),
```

```
p.q1[eltab(Data.tab33, 1, Convert.ToInt16(Data.NA[k, 1]), 0)]);
                        }
                        Data.S = S;
                        //
                        for (int i = Data.nu2; i < p.B.GetLength(0) - Data.ksi2; i++)
                        {
                            double sum = 0;
                            for (int j = 0; j < Data.newnv - 1; j++)
                            {
                                sum = sum + (p.h[j] + p.dh[j]) * p.B[i][j];
                            }
                            hc[i - Data.nu2] = Data.hvh[0] - sum;
                            ucPass01[eltab(Data.tab33, 0, Data.newnv - 1 + i, 1)].Text =
ucPass01[eltab(Data.tab33, 0, Data.newnv - 1 + i, 1)].Text + "\r\n" + "hc" + (Data.newnv - 1
+ i) + "=" + Math.Round(hc[i - Data.nu2], 2);

                        }

                    //напор на выходах НС
                    //НА1
                        if (checkBox6.Checked == true)
                        {
                            {
                                ucPass01[13].Text = ucPass01[13].Text + "\r\n" + "Нвых12=" +
Math.Round((eltab2(Data.vhod, 3, 30, 0) - p.h[eltab(Data.tab33, 1, 8, 0)] -
p.dh[eltab(Data.tab33, 1, 8, 0)]), 2);
                                ucPass01[11].Text = ucPass01[11].Text + "\r\n" + "Нвых11=" +
Math.Round((eltab2(Data.vhod, 3, 31, 0) - p.h[eltab(Data.tab33, 1, 8, 0)] -
p.dh[eltab(Data.tab33, 1, 8, 0)] + p.h[eltab(Data.tab33, 1, 7, 0)] + p.dh[eltab(Data.tab33,
1, 7, 0)] + p.h[eltab(Data.tab33, 1, 6, 0)] + p.dh[eltab(Data.tab33, 1, 6, 0)]), 2);
                            }
                            goto al3;
                        }
                    if (checkBox4.Checked == true)
                    {
                        ucPass01[11].Text = ucPass01[11].Text + "\r\n" + "Нвых11=" +
Math.Round((eltab2(Data.vhod, 3, 30, 0) - p.h[eltab(Data.tab33, 1, 6, 0)] -
p.dh[eltab(Data.tab33, 1, 6, 0)] - p.h[eltab(Data.tab33, 1, 7, 0)] - p.dh[eltab(Data.tab33,
1, 7, 0)] - p.h[eltab(Data.tab33, 1, 10, 0)] - p.dh[eltab(Data.tab33, 1, 10, 0)] +
p.h[eltab(Data.tab33, 1, 20, 0)] + p.dh[eltab(Data.tab33, 1, 20, 0)] + p.h[eltab(Data.tab33,
1, 19, 0)] + p.dh[eltab(Data.tab33, 1, 19, 0)]), 2);
                        ucPass01[13].Text = ucPass01[13].Text + "\r\n" + "Нвых12=" +
Math.Round((eltab2(Data.vhod, 3, 31, 0) - p.h[eltab(Data.tab33, 1, 10, 0)] -
p.dh[eltab(Data.tab33, 1, 10, 0)]), 2);
                    }
                al3:
                    ucPass01[11].BackColor = Color.Aqua;
                        ucPass01[13].BackColor = Color.Aqua;
                    //НС2
                        if (checkBox9.Checked == true)
                        {
                            ucPass01[16].Text = ucPass01[16].Text + "\r\n" + "Нвых22=" +
Math.Round((eltab2(Data.vhod, 3, 29, 0) - p.h[eltab(Data.tab33, 1, 5, 0)] -
p.dh[eltab(Data.tab33, 1, 5, 0)]), 2);
                            ucPass01[15].Text = ucPass01[15].Text + "\r\n" + "Нвых21=" +
Math.Round((eltab2(Data.vhod, 3, 0, 0) - p.h[eltab(Data.tab33, 1, 3, 0)] -
p.dh[eltab(Data.tab33, 1, 3, 0)] + p.h[eltab(Data.tab33, 1, 2, 0)] + p.dh[eltab(Data.tab33,
1, 2, 0)] + p.h[eltab(Data.tab33, 1, 1, 0)] + p.dh[eltab(Data.tab33, 1, 1, 0)]), 2);
                            goto al4;
                        }
                        if (checkBox7.Checked == true)
                        {
                            ucPass01[15].Text = ucPass01[15].Text + "\r\n" + "Нвых21=" +
Math.Round((eltab2(Data.vhod, 3, 0, 0) - p.h[eltab(Data.tab33, 1, 3, 0)] -
p.dh[eltab(Data.tab33, 1, 3, 0)]), 2);
```

```
                        ucPass01[16].Text = ucPass01[16].Text + "\r\n" + "Нвых22=" +
Math.Round((eltab2(Data.vhod, 3, 0, 0) - p.h[eltab(Data.tab33, 1, 3, 0)] -
p.dh[eltab(Data.tab33, 1, 3, 0)]), 2);
                    }
                al4:
        ucPass01[16].BackColor = Color.Aqua;
                    ucPass01[15].BackColor = Color.Aqua;




                //уровнив РЧВ
                    ur5 = Math.Round(((-1 * Convert.ToDouble(dataGridView1[2,
ind(Data.vhod0, 1, 32)].Value) - 1 * Convert.ToDouble(dataGridView1[2, ind(Data.vhod0, 1,
33)].Value)) * 3600 / 27500 + eltab2(Data.vhod0, 3, 33, 1)), 2);
                    Data.ur5 = ur5;

                    dataGridView1[3, ind(Data.vhod0, 1, 32)].Value = ur5;
                    dataGridView1[3, ind(Data.vhod0, 1, 33)].Value = ur5;


                // РЧВ вошло/вышло
                    Data.los1 = -1 * Convert.ToDouble(dataGridView1[2, ind(Data.vhod0, 1,
33)].Value);
                    Data.los2 = -1 * Convert.ToDouble(dataGridView1[2, ind(Data.vhod0, 1,
32)].Value);
                }
            }

            private void checkBox1_CheckedChanged(object sender, EventArgs e)
            {
                //данные узлов
                bool a;
                a = checkBox1.Checked;
                if (a == true)
                {
                    for (int i = Data.ntab1 -Data.nvih- Data.newnvhod + 1; i < Data.ntab1
- Data.newnvhod + 1; i++)
                    {
                        ucPass01[eltab(Data.tab33, 0,i,1)].Visible = true;
                    }
                }
                else
                {
                    for (int i = Data.ntab1 - Data.nvih - Data.newnvhod + 1; i <
Data.ntab1 - Data.newnvhod + 1; i++)
                    {
                        ucPass01[eltab(Data.tab33, 0, i, 1)].Visible = false;
                    }
                }
            }

            private void checkBox2_CheckedChanged(object sender, EventArgs e)
            {
                //данные связей
                bool a;
                a = checkBox2.Checked;
                if (a == true)
                {
                    for (int i = 0; i < nv+Data.nu2; i++)
                    {
                        ucPass01[eltab(Data.tab33, 0, i, 1)].Visible = true;
                    }
```

```
        }
        else
        {
            for (int i = 0; i < nv + Data.nu2; i++)
            {
                ucPass01[eltab(Data.tab33, 0, i, 1)].Visible = false;
            }
        }
    }
```

ВІДОМІСТЬ АТЕСТАЦІЙНОЇ РОБОТИ

Позначення	Найменування	Дод. відомості
	Текстові документи	
1	Пояснювальна записка	79 с.
2	Презентаційний матеріал	20 с.
	Інші документи	
3	Роздруківки програм	12 с.
4	Рецензія	2 с.
5	Відгук керівника	1 с.

					Математичне моделювання та оптимізація квазістаціонарних режимів роботи магістрального водоводу			
Змін	**Арк.**	**Номер докум.**	**Підп.**	**Дата**				
Розроб.		**Євсович О.О.**			**(Тема роботи) Відомість атестаційної роботи**		**Аркуш**	**Аркушів**
Перевір.		**Матвієнко О.І.**						
Н. контр.		**Сидоров М.В.**				**ХНУРЕ Кафедра ПМ**		
Затв.		**Тевяшев А.Д.**						