Правообладатель:

Русяк Иван Григорьевич[а], Дряхлов Руслан Рудольфович[б],
Нефедов Денис Геннадьевич[в]
[а]426063, Российская Федерация, Ижевск, ул. Восточная, д. 75, кв. 9
[б]426069, Российская Федерация, Ижевск, ул. 30 лет Победы, д. 30, ком. 412
[в]426033, Российская Федерация, Ижевск, ул. Нижняя, д. 10, кв. 119

Программа для ЭВМ

Программа для решения задачи Лагранжа с учетом противодавления

Фрагменты исходного текста программы, листов – **10**

Материалы аудиовизуальных отображений,
порождаемых программой для ЭВМ, листов – **2**

Авторы: И.Г. Русяк,
Р.Р. Дряхлов,
Д.Г. Нефедов

Ижевск – 2020

Фрагменты исходного текста программы для ЭВМ

Файл «Form1.cs»

```
using System;
using System.Windows.Forms;
using Снаряд;
using System.IO;
using System.Collections.Generic;
using System.Linq;

namespace Snaryd
{
    public partial class Form1 : Form
    {
        public static string path = "";
        public static List<string> filepath = new List<string> { "", "", "", "", "", "", "", "" };

        double tau = 0.01 * Constans.L_d / Math.Sqrt(2 * Constans.f * Constans.omega / (1 + 1.0 /
3.0 * Constans.omega / Constans.q)
                / Constans.q / Constans.Q) / 2.0;
        public Form1()
        {
            InitializeComponent();

            text_delt.Text = Math.Round(tau * 1000, 5).ToString();
            path = Application.StartupPath;
            filepath[0] = Path.Combine(path, @"t.txt");
            FileInfo filInfo1 = new FileInfo(filepath[0]);
            if (!filInfo1.Exists)
            {
                filInfo1.Create().Close();
            }
            else
            {
                filInfo1.Delete();
                filInfo1.Create().Close();
            }
            filepath[1] = Path.Combine(path, @"v.txt");
            FileInfo filInfo2 = new FileInfo(filepath[1]);
            if (!filInfo2.Exists)
            {
                filInfo2.Create().Close();
            }
            else
            {
                filInfo2.Delete();
                filInfo2.Create().Close();
            }
            filepath[2] = Path.Combine(path, @"p.txt");
            FileInfo filInfo3 = new FileInfo(filepath[2]);
            if (!filInfo3.Exists)
            {
                filInfo3.Create().Close();
            }
            else
            {
                filInfo3.Delete();
                filInfo3.Create().Close();
            }

            filepath[3] = Path.Combine(path, @"p_cn.txt");
            FileInfo filInfo4 = new FileInfo(filepath[3]);
```

```
    if (!filInfo4.Exists)
    {
        filInfo4.Create().Close();
    }
    else
    {
        filInfo4.Delete();
        filInfo4.Create().Close();
    }
    filepath[4] = Path.Combine(path, @"p_km.txt");
    FileInfo filInfo5 = new FileInfo(filepath[4]);
    if (!filInfo5.Exists)
    {
        filInfo5.Create().Close();
    }
    else
    {
        filInfo5.Delete();
        filInfo5.Create().Close();
    }
    filepath[5] = Path.Combine(path, @"l.txt");
    FileInfo filInfo6 = new FileInfo(filepath[5]);
    if (!filInfo6.Exists)
    {
        filInfo6.Create().Close();
    }
    else
    {
        filInfo6.Delete();
        filInfo6.Create().Close();
    }

}


private void button1_Click(object sender, EventArgs e)
{
    int es = 0;
    if (radioButton1.Checked) es = 1;
    else es = 0;
    DrawClear();
    NewDate();
    Constans.NewChet();

    Resalt Param = new Resalt(0, Constans.Pm, Constans.Pm, Constans.Pm, 0, 0,0);
    Draw(Param);
    Decision.Save(Param);
    var col = Decision.Sol(Param, tau,es);
    var col1 = col;
    var vl = col.Select(v => v.V).Max();
    double vl1=vl;
    do
    {
        col = col1;
        Application.DoEvents();
        text_delt.Text = Math.Round(tau * 1000, 5).ToString();
        vl = vl1;
        tau = tau / 2;
        col1 = Decision.Sol(Param, tau,es);
       vl1 = col1.Select(v => v.V).Max();
    }
    while (3*Constans.Eps <Math.Abs(vl1-vl)/vl1);

    foreach (var item in col)
```

```
            {
                Draw(item);
                ConRes(item);
                Decision.Save(item);
            }
        }
        void NewDate()
        {
            Constans.Eps = Convert.ToDouble(text_eps.Text);
            Constans.q = Convert.ToDouble(text_q.Text);
            Constans.d_km = Convert.ToDouble(text_dkm.Text);
            Constans.Pm = Convert.ToInt64(text_pm.Text);
            Constans.d_kn= Convert.ToDouble(text_dkn.Text);
            Constans.L_d = Convert.ToDouble(text_ld.Text);
            Constans.L_km= Convert.ToDouble(text_lk.Text);
            Constans.l_ny = Convert.ToDouble(text_lny.Text);
            Constans.l_ky = Convert.ToDouble(text_lky.Text);
            Constans.N = Convert.ToInt16(text_n.Text);
            Constans.Pn = Convert.ToInt32(text_pn.Text);
            Constans.Cp_v = Convert.ToDouble(text_cpv.Text);
            Constans.Cv_v = Convert.ToDouble(text_cvv.Text);
            Constans.ro_v = Convert.ToDouble(text_rov.Text);
            Constans.Cv_p = Convert.ToDouble(text_cv.Text);
            Constans.Cp_p = Convert.ToDouble(text_cp.Text);
            Constans.al = Convert.ToDouble(text_al.Text);
            Constans.f = Convert.ToInt32(text_f.Text);
        }

        void ConRes(Resalt Param)
        {
            text_v.Text =Math.Round( Param.V,2).ToString();
            text_t.Text = Math.Round(Param.t*1000,2).ToString();
            text_pcp.Text = Math.Round(Param.P/1000000,2).ToString();
            text_pkm.Text = Math.Round(Param.P_km / 1000000, 2).ToString();
            text_pcn.Text = Math.Round(Param.P_cn / 1000000, 2).ToString();
            if (radioButton1.Checked) text_ppr.Text = Math.Round(Param.P_d / 1000000,
2).ToString();
            else text_ppr.Text = "0";
        }

        void DrawClear()
        {
            chart1.Series[0].Points.Clear();
            chart2.Series[0].Points.Clear();
            chart3.Series[0].Points.Clear();
            chart3.Series[1].Points.Clear();
            chart3.Series[2].Points.Clear();
            chart4.Series[0].Points.Clear();
        }
        void Draw(Resalt Param)
        {
            chart1.Series[0].Points.AddXY(Math.Round(Param.t * 1000, 2), Param.V);
            chart2.Series[0].Points.AddXY(Math.Round(Param.t * 1000, 2), Param.l);
            chart3.Series[0].Points.AddXY(Math.Round(Param.t * 1000, 2),
Param.P_km/Math.Pow(10,6));
            chart3.Series[1].Points.AddXY(Math.Round(Param.t * 1000, 2), Param.P / Math.Pow(10,
6));
            chart3.Series[2].Points.AddXY(Math.Round(Param.t * 1000, 2), Param.P_cn / Math.Pow(10,
6));
            if (radioButton1.Checked) chart4.Series[0].Points.AddXY(Math.Round(Param.t * 1000, 2),
Param.P_d / Math.Pow(10, 6));
        }

        private void Form1_Load(object sender, EventArgs e)
```

```
        {

            chart3.ChartAreas[0].CursorX.IsUserEnabled = true;
            chart3.ChartAreas[0].CursorX.IsUserSelectionEnabled = true;
            chart3.ChartAreas[0].AxisX.ScaleView.Zoomable = true;
            chart3.ChartAreas[0].AxisX.ScrollBar.IsPositionedInside = true;


            chart3.ChartAreas[0].CursorY.IsUserEnabled = true;
            chart3.ChartAreas[0].CursorY.IsUserSelectionEnabled = true;
            chart3.ChartAreas[0].AxisY.ScaleView.Zoomable = true;
            chart3.ChartAreas[0].AxisY.ScrollBar.IsPositionedInside = true;
        }

        void Chencg()
        {
            text_v.Text = "0";
            text_t.Text = "0";
            text_pcp.Text = "0";
            text_pkm.Text = "0";
            text_pcn.Text = "0";
            text_ppr.Text = "0";
            tau = 0.01 * Constans.L_d / Math.Sqrt(2 * Constans.f * Constans.omega / (1 + 1.0 / 3.0
* Constans.omega / Constans.q)
                / Constans.q / Constans.Q) / 2.0;
            text_delt.Text = Math.Round(tau * 1000, 5).ToString();
        }
        private void text_dkm_TextChanged(object sender, EventArgs e)
        {
           Chencg();
        }
    }
}

using System;

namespace Snaryd
{
    class Constans
    {
        /// <summary>
        /// Сила пороха, Дж/кг.
        /// </summary>
        public static int f = 1000000;

        /// <summary>
        /// Длина камеры, м.
        /// </summary>
        public static double L_km= 1;

        /// <summary>
        /// Начало уширение камеры, м.
        /// </summary>
        public static double l_ny = 0.5;

        /// <summary>
        /// Конец уширение камеры, м.
        /// </summary>
        public static double l_ky = 0.8;

        /// <summary>
        /// Длина канала ствола, м.
        /// </summary>
        public static double L_d = 5;
```

```
/// <summary>
/// Диаметр камеры орудия, м.
/// </summary>
public static double d_km = 0.3;

/// <summary>
/// Диаметр канала ствола, м.
/// </summary>
public static double d_kn = 0.1;

/// <summary>
/// Площадь камеры орудия,м.
/// </summary>
public static double S_km = Math.PI * d_km * d_km / 4.0;
public static double S_kn = Math.PI * d_kn * d_kn / 4.0;
public static double W_1 = S_km*l_ny;
public static double W_2 = (l_ky - l_ny) * (S_km + S_kn + Math.Sqrt(S_kn * S_km))/3.0;
public static double W_3 = S_kn * (L_km-l_ky);
public static double W_km = W_1 + W_2 + W_3;
public static int N = 200;

public static double Eps=0.001;

public static double Pn = 101325;
public static double ro_v = 1.226;
public static double Cp_v = 1005;
public static double Cv_v = 717;
public static double kv =Cp_v/Cv_v ;

public static double ck = Math.Sqrt(kv*Pn/ro_v);

/// <summary>
/// масса снаряда, кг.
/// </summary>
public static double q = 15;

/// <summary>
/// Максимальное давление, Па.
/// </summary>
public static double Pm = 2500000000;

/// <summary>
/// коволиум пороха, м^3/кг
/// </summary>
public static double al = 0.001;

/// <summary>
/// плотность заряжания, кг/м^3.
/// </summary>
public static double delta = Pm / (f + al * Pm);

public static double Cp_p = 1500;
public static double Cv_p = 1200;

/// <summary>
/// отношение теплоемкости пороховых газов.
/// </summary>
public static double k = (Cp_p/Cv_p)*1.0;


/// <summary>
/// теплоемкости пороховых газов.
/// </summary>
```

```
        public static double Q = k - 1.0;

        /// <summary>
        /// масса заряда, кг.
        /// </summary>
        public static double omega = delta * W_km;

        public static void NewChet()
        {
            S_km = Math.PI * d_km * d_km / 4.0;
            S_kn = Math.PI * d_kn * d_kn / 4.0;
            W_1 = S_km * l_ny;
            W_2 = (l_ky - l_ny) * (S_km + S_kn + Math.Sqrt(S_kn * S_km)) / 3.0;
            W_3 = S_kn * (L_km - l_ky);
            W_km = W_1 + W_2 + W_3;
            kv = Cp_v / Cv_v;
            ck = Math.Sqrt(kv * Pn / ro_v);
            omega = delta * W_km;
            k = (Cp_p / Cv_p) * 1.0;
            delta = Pm / (f + al * Pm);
        }
    }
}
```

Файл «Decision.cs»

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Снаряд;

namespace Snaryd
{
    class Decision:Integral
    {
        public static IEnumerable<Resalt> Sol(Resalt par,double t0,int es)
        {
            var v =  par.V;
            var l = par.l;
            var p = par.P;
            var p_cn = par.P_cn;
            var p_km = par.P_km;
            var p_d = par.P_d;
             var t = par.t;
            do
            {
                v = V_t(v,t0,p_cn,p_d,es);
                l = L_t(v, t0, l);
                var Wcn = W_km + l * S_kn;
                var j0 = J0(l + L_km, Wcn);
                var j1 = J1(l + L_km, Wcn);
                var j2 = J2(l + L_km, Wcn);
                p = (omega * f - (1.0 + omega * j0 / q) * Q * q * v * v / 2.0) / (Wcn - al *
omega);


                p_cn = (p + omega * v * v / Wcn * (j1 - j2 - 1.0 / 2.0 + j0 / 2)) / (1.0 + omega /
q * (j1 - j2));
                p_km = p_cn * (1.0 + omega * j1 / q) + omega * v * v / Wcn * (1.0 / 2.0 - j1);
```

```
                p_d = Pn * (1 + (kv + 1) * kv * v * v / (4 * ck * ck) + kv * v / ck * Math.Sqrt(1 +
Math.Pow((kv + 1) / 4, 2) * v * v / (ck * ck)));

                t = t + t0;
                Resalt newPar = new Resalt(t, p, p_cn, p_km, v, l, p_d);
                yield return newPar;

            }
            while (l < Constans.L_d);

        }

        static double V_t(double v, double t0,double p_cn,double p_d,int es)
        {
            double f = S_kn * (p_cn - p_d * es) / q;
            return v + t0 * S_kn * ((p_cn + t0 * f / 2.0) - (p_d + t0 / 2.0 * f) * es) / q;
        }
        static double L_t(double v, double t0, double l)
        {
            return l + t0 * (v+t0/2.0*v);
        }

        public static void Save(Resalt par)
        {
            SaveData(par.t, "t", Form1.path, FileMode.Append);
            SaveData(par.P, "p", Form1.path, FileMode.Append);
            SaveData(par.P_cn, "p_cn", Form1.path, FileMode.Append);
            SaveData(par.P_km, "p_km", Form1.path, FileMode.Append);
            SaveData(par.l, "l", Form1.path, FileMode.Append);
            SaveData(par.V, "v", Form1.path, FileMode.Append);

        }
        public static void SaveData(double x, string Name, string path, FileMode mode)
        {
            using (FileStream fstream = new FileStream(path + "\\" + Name + ".txt", mode))
            {
                byte[] array =
System.Text.Encoding.Default.GetBytes(x.ToString(System.Globalization.CultureInfo.GetCultureInfo("e
n-US")) + Environment.NewLine);
                fstream.Write(array, 0, array.Length);
            }
        }
    }
}
```

Файл «Integral.cs»

```
using System;

namespace Snaryd
{
    class Integral:Constans
    {
        public static double J0(double Xcn,double Wcn)
        {
            var j = S_kn * S_kn / (Wcn * Wcn * Wcn);
            return j*IntW2S(Xcn);
        }
        public static double J1(double Xcn, double Wcn)
        {
            var j = S_kn * S_kn / (Wcn *  Wcn);
            return j * IntWS(Xcn);
        }
```

```
public static double J2(double Xcn, double Wcn)
{
    var j = S_kn * S_kn / (Wcn * Wcn * Wcn);
    return j * IntS(Xcn);
}
static double IntS(double x)
{
    var h = x / N;
    var s = 0.0;
    for (int i = 0; i < N; i++)
    {
        s += inS (h * i) + inS(h * (i + 1));
    }
    return h * s / 2;
}
static double inS(double x)
{
    return IntWS2(x) * Si(x);
}
static double IntWS2(double x)
{
    if (x <= l_ny) return x * x / 2;
    else
    {
        var h = x / N;
        var s = 0.0;
        for (int i = 0; i < N; i++)
        {
            s += WS(h * i) + WS(h * (i + 1));
        }
        return h * s / 2;
    }
}
static double IntWS(double x)
{
    var h = x / N;
    var s = 0.0;
    for (int i = 0; i < N; i++)
    {
        s += WS(h * i) + WS(h * (i + 1));
    }
    return h * s / 2;
}

static double IntW2S(double x)
{
    var h = x / N;
    var s = 0.0;
    for (int i = 0; i < N; i++)
    {
        s += W2S(h*i)+W2S(h*(i+1));
    }
    return h*s/2;
}
static double W2S(double x)
{
    return Math.Pow(Wi(x),2)/Si(x);
}
static double WS(double x)
{
    return Wi(x) / Si(x);
}
public static double Wi(double x)
{
```

```
            if (x <= l_ny) return S_km * x;
            else if ((x > l_ny) & (x < l_ky)) return W_1 + Sk(x);
            else return W_1 + W_2 + S_kn * (x - l_ky);
        }
        public static double Si(double x)
        {
            if (x <= l_ny) return S_km;
            else if ((x > l_ny) & (x < l_ky)) return S2(x);
            else return S_kn;
        }
        static double Sk(double x)
        {
            return (x - l_ny)*(S_km + S2(x) + Math.Sqrt(S_km * S2(x)))/3;
        }
        static double S2(double x)
        {
            return Math.Pow(Math.Sqrt(S_km)-(Math.Sqrt(S_km)- Math.Sqrt(S_kn))*(x-l_ny)/(l_ky-
l_ny), 2);
        }
    }
}
```

Файл «Resalt.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Снаряд
{
    class Resalt
    {
        /// <summary>
        /// Время
        /// </summary>
        [Formatter("t, мс", 1000)]
        public readonly double t;

        /// <summary>
        /// Скорость снаряда
        /// </summary>
        [Formatter("v, м/с")]
        public readonly double V;

        /// <summary>
        /// Средние давление
        /// </summary>
        [Formatter("P, МПа", 1e-6)]
        public readonly double P;

        /// <summary>
        /// Давление на снаряд
        /// </summary>
        [Formatter("P_cn, МПа", 1e-6)]
        public readonly double P_cn;

        /// <summary>
        /// Давление на дно камеры
        /// </summary>
        [Formatter("P_kn, МПа", 1e-6)]
```

```
        public readonly double P_km;

        /// <summary>
        /// Путь снаряда
        /// </summary>
        [Formatter("L, м")]
        public readonly double l;

        /// <summary>
        /// Противо давление
        /// </summary>
        [Formatter("P_d, м")]
        public readonly double P_d;

        public Resalt(double t, double p, double p_cn, double p_km, double v, double l, double P_d)
        {
            this.t = t;
            this.P = p;
            this.P_cn = p_cn;
            this.P_km = p_km;
            this.V = v;
            this.l = l;
            this.P_d = P_d;
        }
        class FormatterAttribute : Attribute
        {
            public double Coeff { get; }
            public string Name { get; }

            public FormatterAttribute(string postfix, double coeff = 1)
            {
                Coeff = coeff;
                Name = postfix;
            }
        }
    }
}
```

Материалы аудиовизуальных отображений, порождаемых программой для ЭВМ

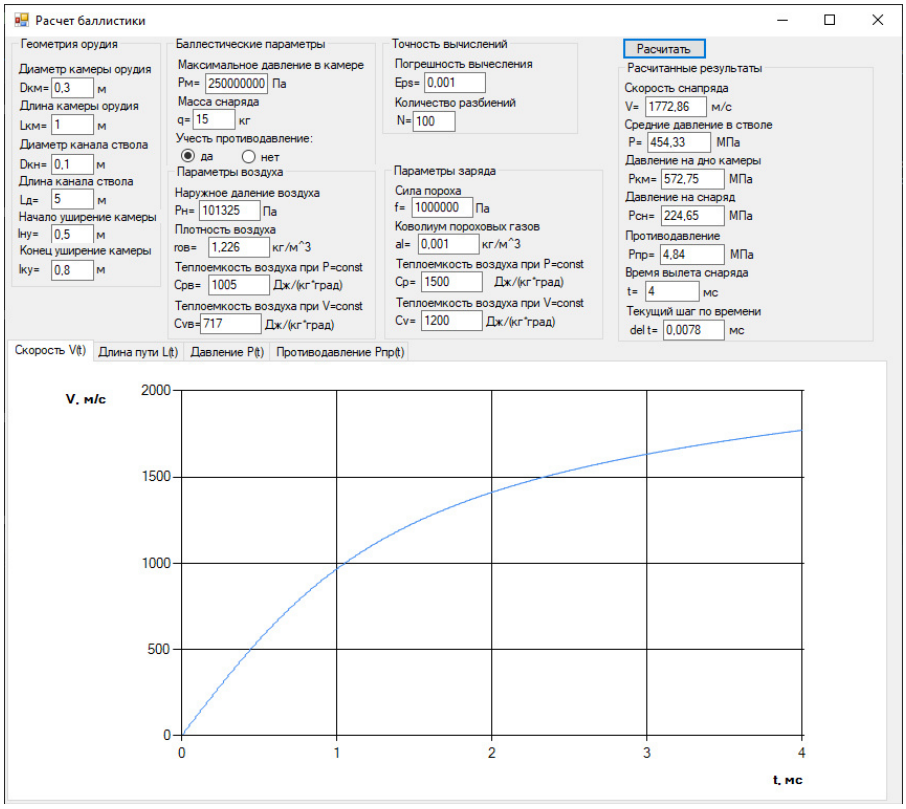

Рисунок 1 – Зависимость изменения скорости снаряда от времени

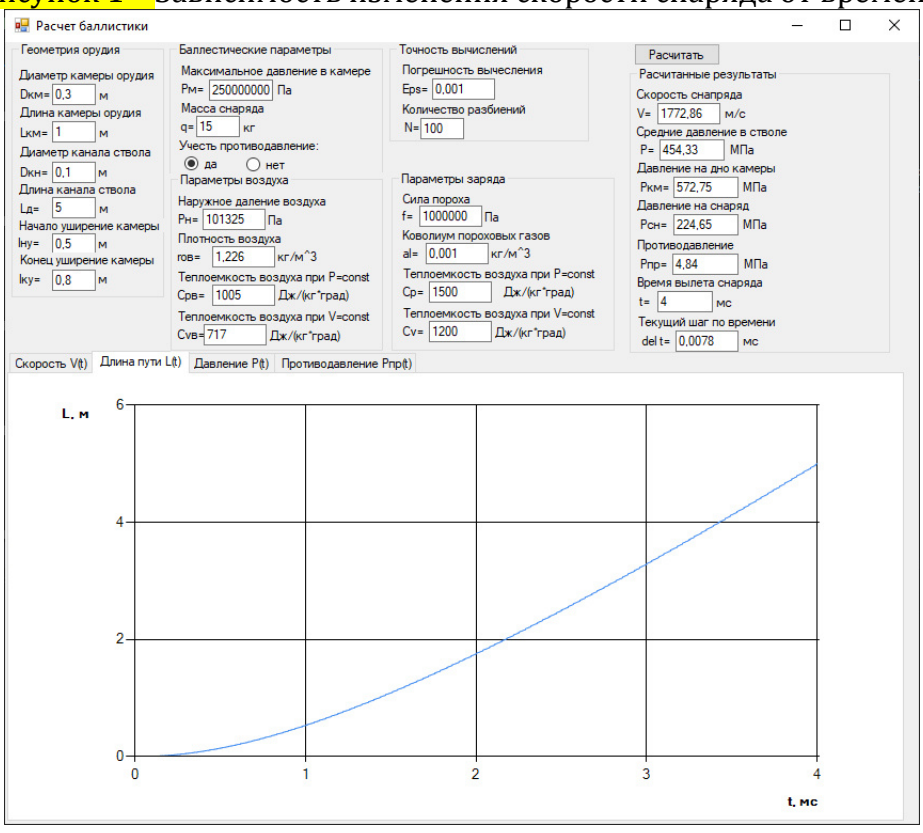

Рисунок 2 – Зависимость изменения движения снаряда от времени

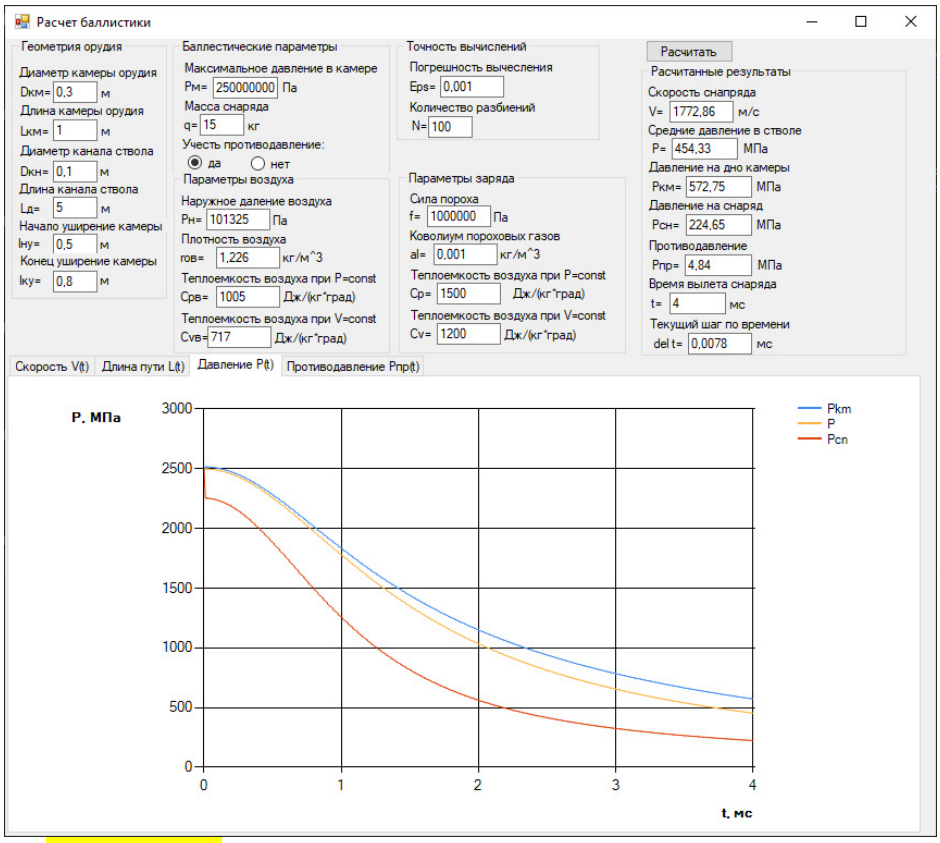

Рисунок 3 – Зависимость изменения давления от времени

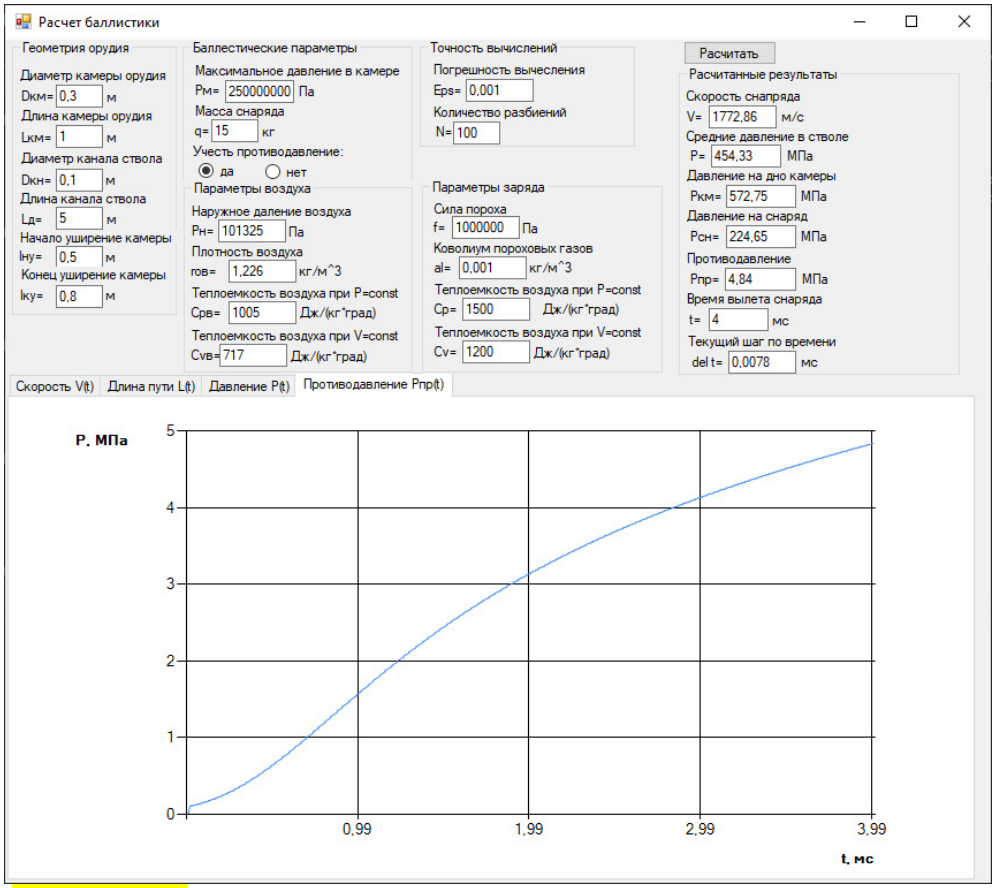

Рисунок 4 – Зависимость изменения противодавления от времени