```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Research.Oslo;
using static System.Math;
using static System.Console;
using mat = MathNet.Numerics.LinearAlgebra.Matrix<double>;
using dmat = MathNet.Numerics.LinearAlgebra.Double.DenseMatrix;
using static MMP.Tools.GnuPlot;
using MMP.Tools;

namespace Ballistics
{
    class Program
    {
        static double scale = 1e-3;
        static double g = 9.80665 * scale;
        static double c = 0.03;

        static Vector fbal(double t, Vector x)
        {
            double y = x[0], v = x[1], a = x[2];
            //t = v[4];

            return new Vector(
                Tan(a),
             -(g * Sin(a) + c * v * v) / (v * Cos(a)),
               -g / (v * v)
               //v * Cos(a)
```

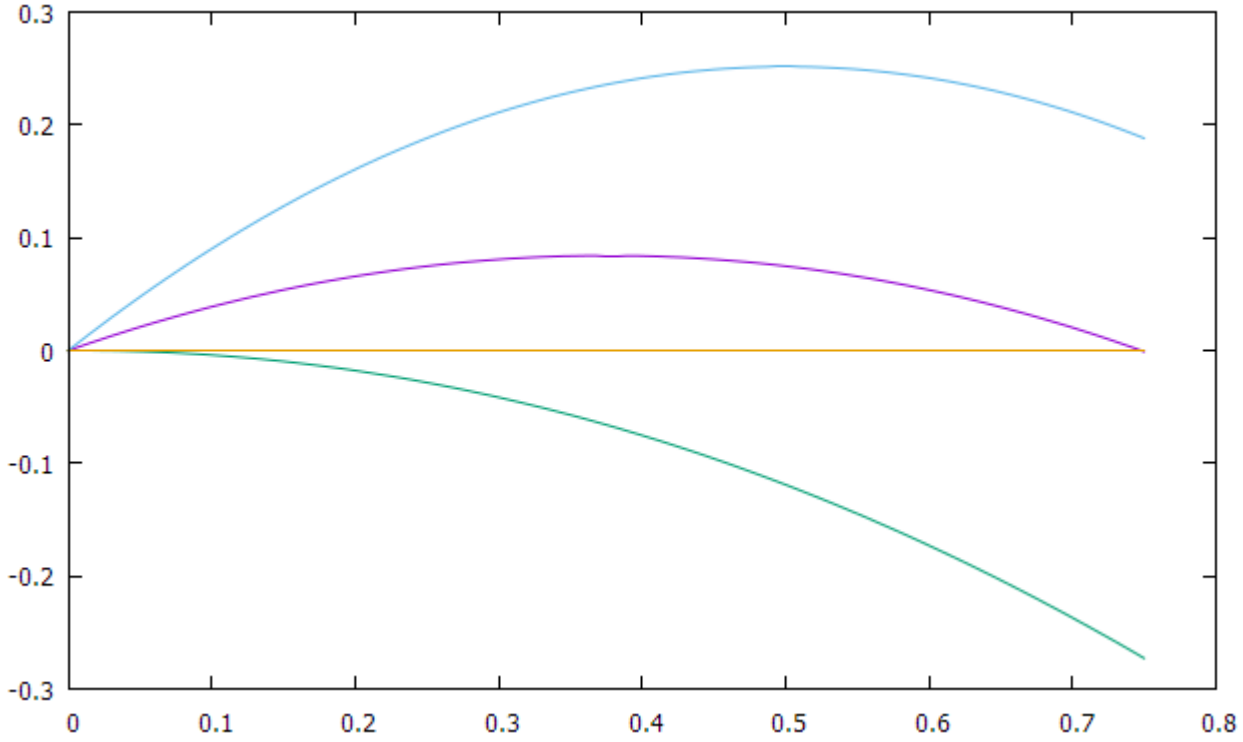
```
            );

        }

        static double Bisection(double v, double L, out List<IEnumerable<SolPoint>> sol, double a1 =
0.01, double a2 = PI / 4, double tol = 0.001, int maxit = 1000)
        {

            Vector y01 = new Vector(0, v, a1);
            Vector y02 = new Vector(0, v, a2);

            Options opt = new Options();
            opt.InitialStep = L / 100;
            double L1 = L + opt.InitialStep / 2;
            var ode1 = Ode.RK45(0, y01, fbal, opt);
            var ode2 = Ode.RK45(0, y02, fbal, opt);
            sol = new List<IEnumerable<SolPoint>>();
            sol.Add(ode1.SolveTo(L1));
            sol.Add(ode2.SolveTo(L1));

            double y1 = sol[0].Last().X[0];
            double y2 = sol[1].Last().X[0];

            if (y1 * y2 > 0)
            {
                WriteLine("No solution!");
                return double.NaN;
            }
            if (y1 == 0)
                return a1;
            if (y2 == 0)
```

```
            return a2;

        double dy = Abs(y2 - y1);
        int it = 0;
        double a = double.NaN;
        IEnumerable<SolPoint> ode = null;
        while (dy > tol && it < maxit)
        {
            a = (a1 + a2) / 2;
            Vector y0 = new Vector(0, v, a);
            ode = Ode.RK45(0, y0, fbal, opt);
            double y = ode.SolveTo(L1).Last().X[0];
            if (y == 0)
                return a;
            if (y1 * y < 0)
            {
                y2 = y;
                a2 = a;
            }
            else
            {
                y1 = y;
                a1 = a;
            }
            it++;
            dy = Abs(y2 - y1);
        }
        sol.Add(ode.SolveTo(L1));
        return a;
    }
```

```
        static void Main(string[] args)
        {
            double v = 100 * scale;
            double L = 750 * scale;
            List<IEnumerable<SolPoint>> sol;
            double a = Bisection(v, L, out sol);
            WriteLine($"Cannon angle {a * 180 / PI}");

            mat data = new dmat(sol[2].Count(), 4);
            var e0 = sol[0].GetEnumerator();
            var e1 = sol[1].GetEnumerator();
            int i = 0;
            foreach (var item in sol[2])
            {
                data[i, 0] = item.T;
                data[i, 1] = item.X[0];
                e0.MoveNext();
                data[i, 2] = e0.Current.X[0];
                e1.MoveNext();
                data[i, 3] = e1.Current.X[0];
                i++;
            }
            var zeroline = dmat.OfArray(new double[,] { { 0, 0 }, { L, 0 } });
            FPlot(data, zeroline);
            WaitForKey();
        }
    }
}


public static void FPlot<T>(params Matrix<T>[] Y)
```

```
    where T : struct, IEquatable<T>, IFormattable
{
    GnuPlot.Write2(Set);
    using (StreamWriter outputFile = new StreamWriter("data"))
    {
        for (int i = 0; i < Y.Length; i++)
        {
            DelimitedWriter.Write(outputFile, Y[i]);
            outputFile.WriteLine();
            outputFile.WriteLine();
            outputFile.WriteLine();
        }

    }

    foreach (var terminal in Terminals)
    {
        GnuPlot.WriteLine2(terminal.ToString());
        string s1, s2;
        for (int k = 0; k < Y.Length; k++)
        {
            for (int i = 1; i < Y[k].ColumnCount; i++)
            {
                if (k==0 && i == 1)
                    s1 = "plot ";
                else
                    s1 = "";
                if (k < Y.Length-1 || i < Y[k].ColumnCount - 1)
                    s2 = @", \";
                else
                    s2 = "";
```

```
                GnuPlot.WriteLine2($"{s1}'data' i {k} u 1:{i + 1} w l t ''{s2}");
            }
        }
        GnupStWr.Flush();
    }
```


0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8