**Математическая модель BallisticLib для расчета траектории полета неуправляемого реактивного снаряда**

Библиотека для .Net Framework 4.7.2:.

Библиотека для .Net Core 3.1: 

Математическая модель вычисляет поведение, скорость и координаты РС на всей траектории полета.

Подключение библиотеки на примере Visual Studio:

1. В обозревателе решений добавить ссылку на библиотеку:

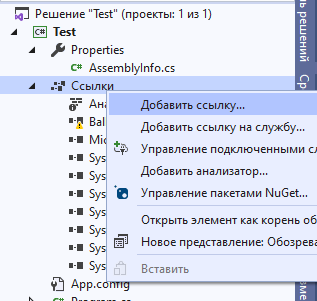


Рис. 1. Добавление в проект ссылки на библиотеку

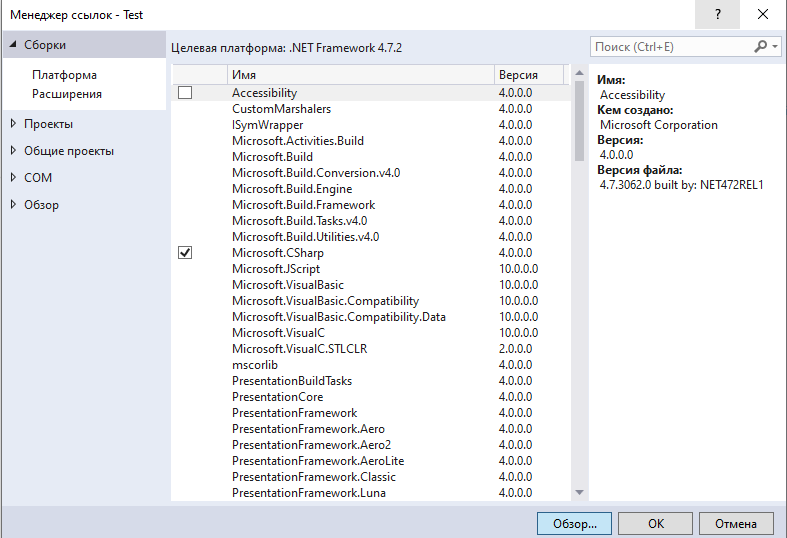


Рис. 2. Окно добавления библиотеки

1. Подключение в исходном файле:

using BallisticLib;

…

BallisticLib.Ballistic ballistic = new Ballistic(

name: "kek",

fuelMass: 23,

massPocketPath: 12.5,

massHeadPath: 34.5,

calibr: 122,

avgValFt: 22500,

timeFuelFire: 1.8,

dlinaNapravl: 5,

usilieStoporen: 100,

koeffForm: 1.3

);

BallisticLib.CalcParams calcParams = new CalcParams(

stepIntegr: 0.05,

timeCalc: 200,

deltaAct: 1,

deltaNotAct: 1

);

double angle = 33.0;

BallisticLib.Calculate calculate = new Calculate();

var res = calculate.Start(ballistic: ballistic, angle: angle, calcParams: calcParams);

1. Описание классов:

В объект класса **ballistic** необходимо передать данные для вычисления:

Структура класса:

public class Ballistic

{

/// <summary>

/// Название эксперимента

/// </summary>

public string Name { get => name; set => name = value; }

/// <summary>

/// Масса топлива

/// </summary>

public double FuelMass { get => fuelMass; set => fuelMass = value; }

/// <summary>

/// Масса обечайки ракетной части

/// </summary>

public double MassPocketPath { get => massPocketPath; set => massPocketPath = value; }

/// <summary>

/// Масса головной части

/// </summary>

public double MassHeadPath { get => massHeadPath; set => massHeadPath = value; }

/// <summary>

/// Калибр

/// </summary>

public double Calibr { get => calibr; set => calibr = value; }

/// <summary>

/// Среднее значение F тяги

/// </summary>

public double AvgValFt { get => avgValFt; set => avgValFt = value; }

/// <summary>

/// Время горения топлива

/// </summary>

public double TimeFuelFire { get => timeFuelFire; set => timeFuelFire = value; }

/// <summary>

/// Длина направляющей

/// </summary>

public double DlinaNapravl { get => dlinaNapravl; set => dlinaNapravl = value; }

/// <summary>

/// Усилие стопорения

/// </summary>

public double UsilieStoporen { get => usilieStoporen; set => usilieStoporen = value; }

/// <summary>

/// Коеффициент формы

/// </summary>

}

В объект класса **calcParams** необходимо передать данные для настройки вычисления

Структура класса:

public class CalcParams

{

/// <summary>

/// Шаг интегрирования

/// </summary>

public double StepIntegr { get => stepIntegr; set => stepIntegr = value; }

/// <summary>

/// Время расчета

/// </summary>

public double TimeCalc { get => timeCalc; set => timeCalc = value; }

/// <summary>

/// Прокуск шагов на акт. участке

/// </summary>

public int DeltaAct { get => deltaAct; set => deltaAct = value; }

/// <summary>

/// Прокуск шагов на не акт. участке

/// </summary>

public int DeltaNotAct { get => deltaNotAct; set => deltaNotAct = value; }

}

В **angle** надо написать угол выстрела.

1. **Выходные данные:**

public struct Result

{

/// <summary>

/// Все Х

/// </summary>

public double[] X;

/// <summary>

/// Все Y

/// </summary>

public double[] Y;

/// <summary>

/// Время полета

/// </summary>

public double[] FlyTimes;

/// <summary>

/// Скорость ракеты

/// </summary>

public double[] FlyV;

/// <summary>

/// Поведение снаряда по оси Х

/// </summary>

public double[] a\_x;

/// <summary>

/// Поведение снаряда по оси У

/// </summary>

public double[] a\_y;

}