Тел: +7 (707) 900 92 67
Почта: saken.yan@yandex.com

10 ЛЕКЦИЯ

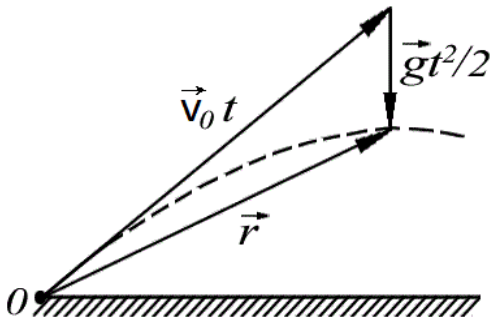
ФИЗИКАЛЫҚ МІНДЕТТЕРДІ КОМПЬЮТЕРЛІК МОДЕЛЬДЕУ.

§10. Горизонтқа бұрышпен лақтырылған дененің қозғалысын ауа кедергісін ескеріп және ескермей модельдеу.

Ауа кедергісін ескермей модельдеу.

Горизонтқа бастапқы кезде α бұрышпен және v_0 жылдамдықпен дене лақтырылған делік. Ауа кедергісін ескермейтін болсақ осы дене үшін Ньютонның екінші заңы мына дифференциалдық теңдеумен сипатталынады.

$$\frac{d^2\mathbf{r}}{dt^2} = \mathbf{g} \tag{10.1}$$



мұндағы $\mathbf{r}$ – координата басынан дененің t уақыттағы орнына жүргізілген радиус векторы және $\mathbf{g}$ - ауырлық күшінің үдеуі.

$$\frac{d\mathbf{r}}{dt} = \mathbf{v} \tag{10.2}$$

екенің ескерсек (10.1) бірінші ретті дифференциалдық теңдеулер жүйесі түрінде жаза аламыз:

$$\begin{cases} \frac{d\mathbf{v}}{dt} = \mathbf{g} \\ \frac{d\mathbf{r}}{dt} = \mathbf{v} \end{cases} \tag{10.3}$$

Енді координат осьтерін енгізіп, жылдамдықты, үдеуді және радиус векторды проекциялар бойынша жазсақ.

OX үшін:

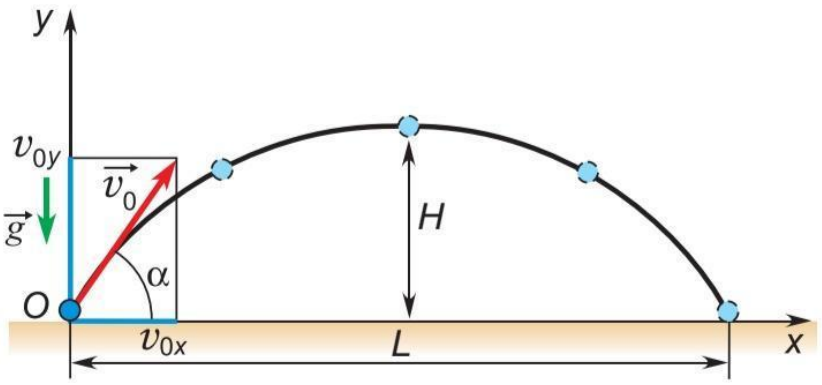
$$\begin{cases} \frac{dx}{dt} = v_x \\ \frac{dv_x}{dt} = 0 \end{cases} \tag{10.4}$$

OY үшін:

$$\begin{cases} \frac{dy}{dt} = v_y \\ \frac{dv_y}{dt} = -\mathrm{g} \end{cases} \tag{10.5}$$

Бастапқы жылдамдықты v_0 вертикаль және горизонталь компоненттерге жіктеп жазсақ:

$$\begin{cases} v_{0x} = v_0 cos\alpha \\ v_{0y} = v_0 sin\alpha \end{cases} \tag{10.6}$$



Енді туындыларды Эйлердің схемасы арқылы жазайық, яғни мына түрде:

$$\frac{df}{dt} = \frac{f_{i+1} - f_i}{\Delta t} = \frac{f_{i+1} - f_i}{h} \tag{10.7}$$

Сонда (10.4) және (10.5) Эйлер схемасы түрінде жазақ:

OX үшін:

$$\begin{cases} x_{i+1} = x_i + v_{x_{i+1}} \Delta t \\ v_{x_{i+1}} = v_{x_i} = v_0 cos\alpha = const \end{cases} \tag{10.8}$$

OY үшін:

$$\begin{cases} y_{i+1} = y_i + v_{y_{i+1}} \Delta t \\ v_{y_{i+1}} = v_{y_i} - \mathrm{g}\Delta t \end{cases} \tag{10.9}$$

Мұнда биіктік артқан сайын жылдамдықтың y компонентісі яғни v_{y_i} азаяды, бірақ оң болып қалады. Максималды биіктікке жеткеннен соң ғана ол теріс болады. y координатасының өзі дененің жерге құлауы мүмкін емес мағынасында теріс болмайды.

Ауа кедергісін ескере отырып модельдеу.

Дененің бастапқы кездегі жылдамдығы көп болғанда ауа кедергісі қозғалысты айтарлықтай өзгерте алады. Ауаның кедергі күшін мына түрде деп ескерсек:

$$\overrightarrow{F_{\text{сопр}}} = k_1 \vec{\mathbf{v}} + k_2 \vec{\mathbf{v}}^2$$

$\overrightarrow{F_{\text{сопр}}} = F_x \overrightarrow{\mathbf{e_x}} + F_y \overrightarrow{\mathbf{e_y}}$ болғандықтан және ауаның кедергі күшін ескере отырып (10.4) және (10.5) өрнектерін қайта жазсақ:

OX үшін:

$$\begin{cases} \dfrac{dx}{dt} = v_x \\ \dfrac{dv_x}{dt} = \dfrac{F_x}{m} = -\dfrac{k_1 + k_2\sqrt{(v_x^2 + v_y^2)}}{m} v_x \end{cases} \tag{10.10}$$

OY үшін:

$$\begin{cases} \dfrac{dy}{dt} = v_y \\ \dfrac{dv_y}{dt} = -\mathrm{g} - \dfrac{k_1 + k_2\sqrt{(v_x^2 + v_y^2)}}{m} v_y \end{cases} \tag{10.11}$$

Сондықтан (10.8) және (10.9) итерациялау алгоритмі келесі түрге ие болады:

OX үшін:

$$\begin{cases} x_{i+1} = x_i + v_{x_{i+1}}\Delta t \\ v_{x_{i+1}} = v_{x_i} + \left(-\dfrac{k_1 + k_2\sqrt{\left(v_{x_i}^2 + v_{y_i}^2\right)}}{m} v_{x_i} \right)\Delta t \end{cases} \tag{6.12}$$

OY үшін:

$$\begin{cases} y_{i+1} = y_i + v_{y_{i+1}}\Delta t \\ v_{y_{i+1}} = v_{y_i} - \mathrm{g}\Delta t + \left(-\dfrac{k_1 + k_2\sqrt{\left(v_{x_i}^2 + v_{y_i}^2\right)}}{m} v_{y_i} \right)\Delta t \end{cases} \tag{6.13}$$

Мысал.

Дене көкжиекке α бұрышпен және бастапқы v_0 жылдамдықпен лақтырылған. Ауа кедергісін есепкерілмейді. L дененің ұшу қашықтығы және H дененің көтерілу биіктігі уақыттан қалай тәуелді екенін модельдеу арқылы табыңыз. Дененің ұшу траекториясын сызыңыз, яғни y = y (x) графигін салыңыз.

Алдымен бағдарламаның жалпы құрылымы көрсетілген. Содан кейін әр әдістің орындалуы беріледі.

```
class Program
{
        //Статичный класс
    public static class Checker
    {
      //Статичный метод класса проверка на приземление
      public static bool IsLanded(float x, float y)
      {/* Реализация метода на след стр. */}
    }
```

```
    //Класс брошенное тело
    public class ThrownBody
    {
        //Координаты и проекций скорости тела.
        private List<float> x;  private List<float> vx;
        private List<float> y;  private List<float> vy;

        //Конструктор.
        public ThrownBody(float x, float y, float v, float deg)
        {/* Реализация конструктора на след стр. */}

        //Ограничитель движения.
        public bool IsLanded()
        {/* Реализация метода на след стр. */}

        //Итератор движения.
        public void Next(float dt)
        {/* Реализация метода на след стр. */}

        //Брасок.
        public void Throw(float dt)
        {/* Реализация метода на след стр. */}

        //Вывод на экран.
        public void Show()
        {/* Реализация метода на след стр. */}
         }

    //Статичный метод класса Program
    static void Main(string[] args)
    {
        ThrownBody Body1 = new ThrownBody(0, 0, 10, 45);

        Body1.Throw(0.01f);
        Body1.Show();

        Console.ReadKey();
    }
}//Program
```

Реализация каждого метода.

```
//Статичный метод класса проверка на приземление
public static bool IsLanded(float x, float y)
{
        if (y < 0) return  true;
            else return false;
}
```

```
//Конструктор.
public ThrownBody(float x, float y, float v, float deg)
{
        if (deg <=0 || deg >= 180)
        throw new Exception("Creating object of flying body is failed due to angle");

        deg = (float)((deg * Math.PI)/180);

        this.x = new List<float>();  this.vx = new List<float>();
        this.y = new List<float>();  this.vy = new List<float>();
```

```
        this.x.Add(x);  this.vx.Add((float)(v * Math.Cos(deg)));
        this.y.Add(y);  this.vy.Add((float)(v * Math.Sin(deg)));
}
```

```
//Ограничитель движения.
public bool IsLanded()
{
        if (x.Count == 0 || vx.Count == 0) return true;
        if (y.Count == 0 || vy.Count == 0) return true;
        return Checker.IsLanded(x[x.Count - 1], y[y.Count - 1]);
}
```

```
//Итератор движения.
public void Next(float dt)
{
        if (object.ReferenceEquals(this.x, null)) return;
        if (object.ReferenceEquals(this.y, null)) return;

        if (object.ReferenceEquals(this.vx, null)) return;
        if (object.ReferenceEquals(this.vy, null)) return;

        if (x.Count == 0 || vx.Count == 0) return;
        if (y.Count == 0 || vy.Count == 0) return;

        int Count = x.Count;

        if (Count!= y.Count || vx.Count != Count || vy.Count != Count)
        { Console.WriteLine("Код ошибки: 001. Итерация прервано..."); return; }

        //Начало перемещения.
        this.vx.Add(this.vx[this.vx.Count - 1]);
        this.vy.Add(this.vy[this.vy.Count - 1] - 9.8f*dt);

        this.x.Add(this.x[this.x.Count - 1] + this.vx[this.vx.Count - 1] * dt);
        this.y.Add(this.y[this.y.Count - 1] + this.vy[this.vy.Count - 1] * dt);
}
```

```
//Брасок.
public void Throw(float dt)
{
        Console.WriteLine("Flight is started...");

        while (!IsLanded())
            Next(dt);

        Console.WriteLine("Flight is completed!");
}
```

```
//Вывод на экран.
public void Show()
{
   if (this.x.Count != this.y.Count)
   { Console.WriteLine("Not possible to show data!"); }

   for( int i = 0; i < this.x.Count; i ++ )
   Console.WriteLine("x = {0};y = {1}", x[i], y[i]);
}
```

№10 зертханалық сабақтың тапсырмалары.

№10 дәрістердің соңында көрсетілген кодты негізге алыңыз.

1. **ThrownBody** классы ішіне XMax() және YMax() деген әдістерді қосыңыздар. Бұл әдіс x және y тізімдері ішінен максималды мәндерін қайтаруы және бұл мәндерді консолға шығаруы тиіс.

2. **Initialiser** деген статикалық класты құрыңыз. Бұл класстың X_ZERO, Y_ZERO, ANGLE және SPEED деген 4 статикалық float типтес жолдары болуы керек және олардың бастапқы мәндері сәйкесінше: X_ZERO = 0, Y_ZERO = 0, ANGLE = 45, SPEED = 10.

Сонымен қатар **Initialiser** класының ішінде InPut() деген әдісі бар болсын. Бұл әдіс нәтиже ретінде true (егер еңгізу дұрыс болса) false (егер бұрыс болса) қайтарады. Бұл әдіс арқылы программаны қолданушы X_ZERO, Y_ZERO, ANGLE және SPEED мәндерін еңгізе алуы керек. Сонымен қатар, бұл әдіс қауіпсіз еңгізуді қамтамассыз етуі керек.

3. **ThrownBody** классы ішіне Write() деген әдісті қосыңыз. Бұл әдіс дененің траекториясын яғни $y = y(x)$ функциясын Traj.dat деген жұмыс үстеліндегі файлға жазуы тиіс. Файлға жазылған деректер арқылы Excel немесе Origine Lab программаларын қолданып $y = y(x)$ функциясының графиктерін тұрғызыңыз.

4. 4.1. Санақтар (enum) туралы оқыңыз. Program классы ішінде WINDAGE деп аталатын Санақты құрыңыз және бұл санақ ON және OFF деп аталатын екі тұрақтылардан тұруы керек.

4.2. **ThrownBody** классы ішінде WINDAGE типті аты Windage болатын private модификаторы арқылы белгіленген кластың жолын жазыңыз.

4.3. **ThrownBody** классы ішінде дененің массасына арналған кластың жолын жазыңыз. Оның бастапқы кездегі мәні 1кг.

4.4. medium деген статикалық класты құрыңыз. Оның k1 деген жолы болуы тиіс. Бастапқыдағы мәні k1=0.

4.4. Создайте статичный класс medium с одним статичным полем k1 и по умолчанию присвойте значение 0.

4.5. **ThrownBody** классы үшін төмендегідей тағы бір конструкторды жазыңыз.

```
public ThrownBody(float x, float y, float v, float deg, WINDAGE Win) :                    this(x,
y, v, deg)
{
    this.Windage = Win;
}
```

4.6. Жылдамдықтардың проекцияларын есептеу алгоритмін жеке әдіске шығарыңыз.

```
public void NextSpeed(float dt)
{...}
```

бұл әдіс Windage қабылдайтын мәніне байланысты Жылдамдықтардың проекцияларын ауаның кедергісін ескеріп не ескермей есептеуі тиіс.

4.7. Main әдісі ішінде medium.k1 = 0.5f деп орнатыныз. Есептеуді ауаның кедергісін ескере отырып және ескермей есептеңіздер. Дененің ұшу биіктігі мен ұшу қашықтығы екі жағдайда қаншалықты өзгеше екенің анықтаңыз. Осы екі қозғалыс үшін траектория графигін тұрғызыңыз. Бастапқы шарттар: v_0=10, alfa=45, x_0=0, y_0=0.