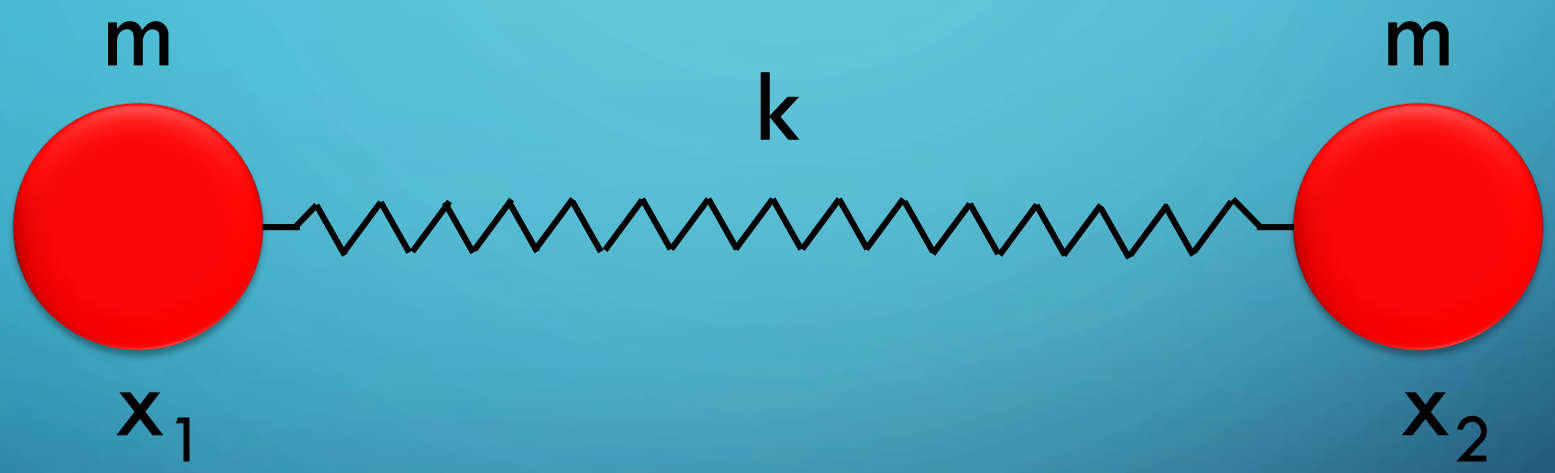
UKŁAD DWÓCH PUNKTÓW MATERIALNYCH POŁĄCZONYCH SPRĘŻYNĄ

MODEL MATEMATYCZNY UKŁADU

SCHEMAT UKŁADU



Energia kinetyczna układu:

$$T = \frac{1}{2} m(\dot{x}_1^2 + \dot{x}_2^2)$$

Energia potencjalna układu:

$$U = \frac{1}{2} k(x_2 - x_1 - l_0)^2$$

Funkcja Lagrange'a:

$$L = T - U$$

$$L = \frac{1}{2} m(\dot{x}_1^2 + \dot{x}_2^2) - \frac{1}{2} k(x_2 - x_1 - l_0)^2$$

Równania Eulera-Lagrange'a

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}_1} - \frac{\partial L}{\partial x_1} = 0$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}_2} - \frac{\partial L}{\partial x_2} = 0$$

Dla lagranżjanu opisującego rozważany układ mechaniczny równania E – L przybierają poniższą postać:

$$\ddot{x}_1 + \omega_0^2(x_1 - x_2 + l_0) = 0$$

$$\ddot{x}_2 + \omega_0^2(x_2 - x_1 - l_0) = 0$$

$$\omega_0^2 = \frac{k}{m}$$

Podamy rozwiązanie układu równań Eulera-Lagrange'a dla dwóch wybranych zestawów warunków początkowych

$$x_1(0) = 0 \;\; \dot{x}_1(0) = 0 \;\; x_2(0) = x_0 \;\; \dot{x}_2(0) = 0$$

$$x_1(0) = 0 \;\; \dot{x}_1(0) = 0 \;\; x_2(0) = l_0 \;\; \dot{x}_2(0) = v_0$$

Rozwiązanie dla pierwszego zestawu warunków początkowych

$$x_1 = \frac{x_0 - l_0}{2}\left[1 - cos\sqrt{2}\omega_0 t\right]$$

$$x_2 = x_0 - \frac{x_0 - l_0}{2}\left[1 - cos\sqrt{2}\omega_0 t\right]$$

Rozwiązanie dla drugiego zestawu warunków początkowych

$$x_1 = v_0 t - \frac{v_0}{\omega_0} sin\omega_0 t$$

$$x_2 = l_0 + \frac{v_0}{\omega_0} sin\omega_0 t$$

VPYTHON

VPython to biblioteka stanowiąca rozszerzenie języka Python o możliwość programowania grafiki 3D. Pozwala tworzyć obiekty reprezentujące takie obiekty graficzne jak sfery, cylindry czy prostopadłościany, renderować je w oknie oraz tworzyć animacje i manipulować nimi w interaktywny sposób. Biblioteka jest zorientowana na maksymalne uproszczenie programowania, co pozwala skupić się na stronie obliczeniowej tworzonej wizualizacji.

VPYTHON – PRZYKŁADOWE WBUDOWANE OBIEKTY

sphere (sfera)

obj=sphere(pos=(0,0,-1),radius=0.5,color=color.yellow)

box (prostopadłościan)

obj=box(pos=(0,0,-1),width=1,height=1,length=1,color=color.yellow,axis=(1,0,0))

VPYTHON – PRZYKŁADOWE WBUDOWANE OBIEKTY

cylinder (cylinder)

obj=cylinder(pos=(-1,0,0),axis=(1,0,0),radius=0.5,color=color.yellow)

helix (spirala)

obj=helix(pos=(-1,0,0),axis=(1,0,0),radius=0.5,coils=5,color=color.orange)

PROGRAM

Celem programu jest wizualizacja ruchu dwóch punktów materialnych połączonych sprężyną. Ruch generowany jest przez wychylenie punktu materialnego (reprezentowanego przez sferę) z położenia równowagi lub/i nadanie mu prędkości początkowej.

PROGRAM-KOD

Program składa się z czterech klas. Poniżej zostaną przedstawione najistotniejsze elementy kodu.

```
def Force(self):

    d = self.m2.sphere.pos - self.m1.sphere.pos
    l = mag(d)
    self.w = norm(d)
    Fs = (l - self.lo) * self.k * self.w
    self.m1.F += Fs
    self.m2.F -= Fs
```

```
# we define a elastic force

# the difference (vector) positions of material points
# the magnitude of a vector d
# a unit vector in the direction of the vector d
# a elastic force (vector)
# a force acting out first material point
# a force acting out second material point
```

PROGRAM - KOD

```
self.frame = frame()                    # frame() - group objects together to make a composite object that can be moved as though
self.frame.pos = self.m1.sphere.pos     # be moved as though it were a single object
self.frame.axis = self.m2.sphere.pos - self.m1.sphere.pos
s = 10                                  # parameters of a spring
dx = 0.01
self.helix = curve(frame=self.frame, radius=0.01, color=color) # the curve object displays straight lines between points, and if the points
for x in arange(0, 10 + dx/2.0, dx):    # are sufficiently close together you get the appearance of a smooth curve
    self.helix.append( pos=(x, 0.03*sin(s*x), 0.03*cos(s*x)) )    # arange()- function to build a vector containing an arithmetic progression
self.count = len(self.helix.pos)        # len() - return the length (the number of items) of an object
```

PROGRAM - KOD

```
self.sphere = sphere(radius=0.15, color=color)          # sphere() - an internal object in vpython which creates a sphere
self.sphere.mass = self                                 # we need this to access the mass in the mouse_action loop
self.m = float(m)
self.sphere.pos = vector(pos)                           # vector() - an internal object in vpython which creates a vector
self.fixed = 0.0                                        # we need this to provide interaction with mouse
self.pickable = 1.0
self.v = vector(0.0, 0.0, 0.0)                          # the initial vector of the velocity
self.F = vector(0.0, 0.0, 0.0)                          # the initial vector of the force
```

PROGRAM - KOD

```
ker = Kernel(tstep=0.04)

mass1 = Material_point(m=0.1, pos=(0.0, 0.0, 0.0))

#mass1 = Material_point(m=0.1, pos=(1.2, 0.0, 0.0))

mass2 = Material_point(m=0.1, pos=(0.5, 0.0, 0.0))

ker.addMaterial_point(mass1)

ker.addMaterial_point(mass2)

spring = Spring(m1=mass1, m2=mass2,k=0.1,lo=0.5)

ker.addSpring(spring)

ker.loop()
```

BIBLIOGRAFIA

- https://sage2.icse.us.edu.pl/home/pub/558/

- http://vpython.org

- http://guigui.developpez.com/cours/python/vpython/en/?page=windowseventfile