Układ dwóch punktów materialnych połączonych sprężyną.

A. Szumny T.J.P.

Luty 2017

1 Wstęp

Celem programu jest wizualizacja ruchu układu dwóch punktów materialnych połączonych sprężyną. Ruch jest generowany przez wychylenie punktu materialnego z położenia równowagi lub/i przez nadanie prędkości początkowej. Program został napisany z wykorzystaniem Visual Python. VPython to biblioteka stanowiąca rozszerzenie języka Python o możliwość programowania grafiki 3D. Pozwala tworzyć obiekty reprezentujące takie struktury 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.

2 Model matematyczny układu

Równania ruchu układu zostały wyznaczone przy użyciu zasady Hamiltona. Zasada ta mówi że dla każdego układu mechanicznego istnieje funkcja charakteryzująca ten układ (zwana funkcją Lagrange'a), taka że ruch układu spełnia następujący warunek. Niech w chwilach $t = t_1$ i $t = t_2$ układ ma określone położenia scharakteryzowane przez dwa zbiory wartości współrzędnych uogólnionych $q^{(1)}$ i $q^{(2)}$. Wtedy między tymi położeniami układ porusza się tak , że całka z funkcji Lagrange'a po czasie przyjmuje wartość ekstremalną. Całkę tą nazywamy działaniem układu. Wariując działanie i przyrównując otrzymany wynik do zera (warunek konieczny istnienia ekstremum dla funkcjonału) wprowadzamy równania Eulera-Lagrange'a II rodzaju za pomocą których wyznaczymy ruch układu.

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

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

Funkcja Lagrange'a jest różnicą energii kinetycznej i potencjalnej układu , dla rozważanego układu przyjmuje postać:

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

Dla lagranżjanu opisującego rozważany układ mechaniczny równania Eulera–Lagrange'a przybierają poniższą postać:

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

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

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

Powyższy układ równań różniczkowych drugiego rzędu został rozwiązany metodą podstawiania dla dwóch zestawów warunków początkowych odpowiadających możliwym generacją ruchu w programie. Poniżej podajemy warunki początkowe z rozwiązaniem.

Rozwiązanie dla pierwszego zestawu warunków początkowych:

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

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

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

Rozwiązanie dla drugiego zestawu warunków początkowych:

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

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

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

3 Kod programu

Program składa się z krótkiego opisu działania programu , czterech klas tworzących elementy graficzne i animację ruchu układu , oraz kodu wykonującego program. Zamieszczamy najważniejsze części kodu wraz z komentarzami(zaznaczonymi na niebiesko).

we define a elastic force

def Force(self):

the difference (vector) positions of material points

d = self.m2.sphere.pos - self.m1.sphere.pos

the magnitude of a vector d

l = mag(d)

a unit vector in the direction of the vector d

self.w = norm(d)

a elastic force (vector)

Fs = (l - self.lo) * self.k * self.w

a force acting out first material point

self.m1.F += Fs

a force acting out second material point

self.m2.F -= Fs

frame() - group objects together to make a composite object that can be moved as though be moved as though it were a single object

self.frame = frame()

self.frame.pos = self.m1.sphere.pos

parameters of a spring

s = 10
dx = 0.01

the curve object displays straight lines between points, and if the points are sufficiently close together you get the appearance of a smooth curve

self.helix = curve(frame=self.frame, radius=0.01, color=color)
for x in arange(0, 10 + dx/2.0, dx):
self.helix.append(pos=(x, 0.03*sin(s*x), 0.03*cos(s*x)))

len() - return the length (the number of items) of an object

self.count = len(self.helix.pos)

sphere() - an internal object in vpython which creates a sphere

self.sphere = sphere(radius=0.15, color=color)

we need this to access the mass in the mouseaction loop

self.sphere.mass = self

self.m = float(m)

vector() - an internal object in vpython which creates a vector

self.sphere.pos = vector(pos)

we need this to provide interaction with mouse
self.fixed = 0.0
self.pickable = 1.0

4 Źródła

Pisząc kod programu korzystaliśmy z poniższych źródeł:

http://sage2.icse.us.edu.pl/home/pub/558/

http://vpython.org

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