Drgająca struna

Vibrating String

Maria Budziło, Marcin Kubański, Hiacynta Stypuła

11.01.2022

Politechnika Krakowska
Wydział Inżynierii
Materiałowej i Fizyki

Plan prezentacji

- Wstęp. Opis problemu
- Równanie hiperboliczne fali
- Rozwiązanie numeryczne
- Algorytm kroku czasowego
- Implementacja. Rozwiązanie problemu
- Podsumowanie
- Bibliografia

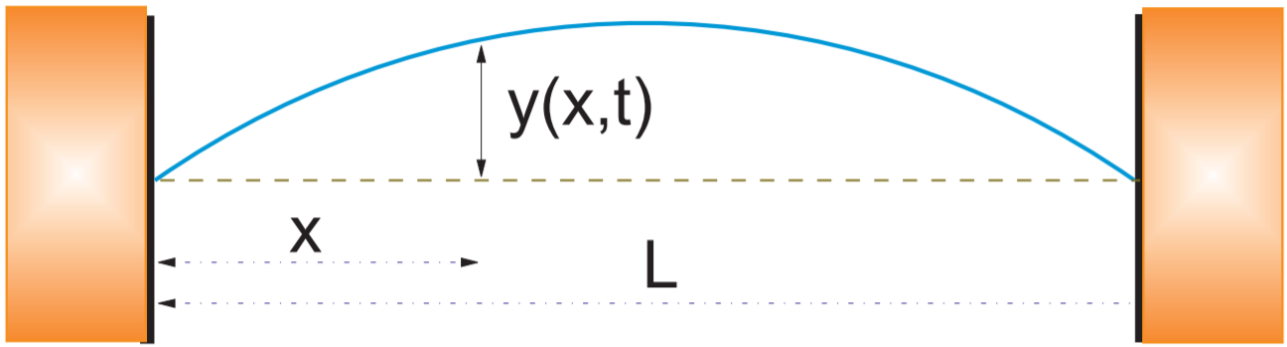
Wstęp. Opis problemu

Przypomnijmy sobie demonstrację z fizyki elementarnej, w której lina przywiązana na obu końcach jest 'delikatnie' szarpana w jednym miejscu i obserwuje się, że wzdłuż niej rozchodzi się impuls. Podobnie, jeśli struna ma jeden koniec wolny i potrząśniemy nią w odpowiedni sposób, powstanie wzór fali stojącej, w której węzły pozostają na swoim miejscu, a strzałki fali poruszają się tylko w górę i w dół, osiągając maksima amplitudy.

W tej prezentacji opracujemy model propagacji fali na strunie.

Równanie hiperboliczne fali

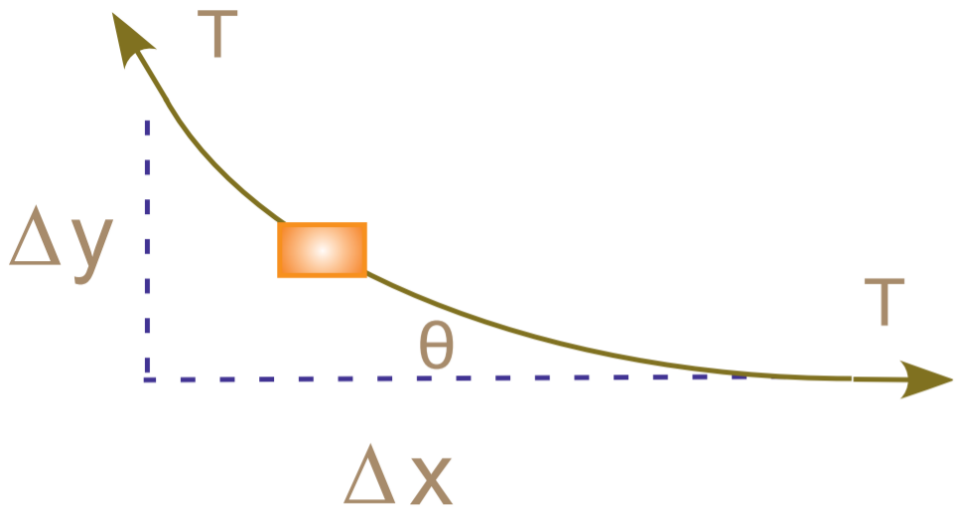
Rozważmy strunę o długości L, przywiązanej z dwóch końców, tak jak rysunek poniżej. Struna ma stałą gęstość ρ na jednostkę długości, stałe naprężenie T i nie ma żadnych sił tarcia. Ustalamy, że naprężenie jest tak duże, że możemy zaniedbać oddziaływanie. grawitacyjne. Zakładamy, że przemieszczenie struny y(x, t) od położenia spoczynkowego następuje tylko w kierunku pionowym i jest funkcją poziomego położenia wzdłuż struny x i czasu t.



Rysunek: Struna z przywiązanymi końcami.

Równanie hiperboliczne fali

Aby otrzymać proste liniowe równanie ruchu, zakładamy, że względne przemieszczenie struny $\frac{y(x,t)}{L}$ i nachylenie $\frac{\partial y}{\partial x}$ są małe.



Rysunek: Struna z przywiązanymi końcami.

Równanie hiperboliczne fali

$$\sum F_y = \rho \cdot \Delta x \cdot \frac{\partial^2 y}{\partial t^2}$$

$$= T \sin\theta(x + \Delta x) - T \sin\theta(x)$$

$$= T \frac{\partial y}{\partial x} \,|_{x+\Delta x} - \frac{\partial y}{\partial x}| \,|_x \approx T \frac{\partial^2 y}{\partial x^2}$$

$$\Rightarrow \frac{\partial^2 y(x,t)}{\partial x^2} = \frac{1}{c^2} \frac{\partial^2 y(x,t)}{\partial t^2}, c = \sqrt{\frac{T}{\rho}}$$

Równanie hiperboliczne fali

Warunki początkowe

$$y(x, t = 0) = \begin{cases} 1.25x/L, & x \leqslant 0.8L \\ (5 - 5x/L) & x > 0.8L \end{cases}$$

$$\frac{\partial y}{\partial t}(x, t = 0) = 0$$

Warunki brzegowe

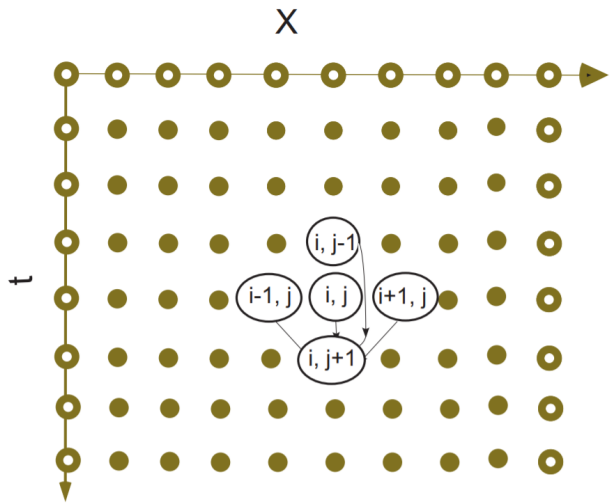
$$y(0, t) \equiv 0,$$

$$y(L, t) \equiv 0$$

Rozwiązanie numeryczne

- $y(x,t) = X(x)T(t)$
- $\frac{d^2T(t)}{dt^2} + \omega^2 T(t) = 0, \quad \frac{d^2X(x)}{dx^2} + k^2 X(x) = 0, \quad k \stackrel{def}{=} \frac{\omega}{c}$
- $X(x=0,t) = X(x=l,t) = 0$
- $\Rightarrow X_n(x) = A_n \sin k_n x, \quad k_n = \frac{\pi(n+1)}{L}, \quad n = 0,1,....$
- $T_n(t) = C_n \sin \omega_n t + D_n \cos \omega_n t, \;\; \omega_n = nck_0 = n\frac{2\pi c}{L}$
- $y_n(x,t) = \sin k_n x \cos \omega_n t, \quad n = 0,1....$
- $y(x,t) = \sum_{n=0}^{\infty} B_n \sin k_n x \cos \omega_n t$
- $y(x,t=0) = \sum_{n}^{\infty} B_n \sin nk_0 x$
- $B_m = 6.25 \cdot \frac{\sin(0.8m\pi)}{m^2\pi^2}$

Algorytm kroku czasowego

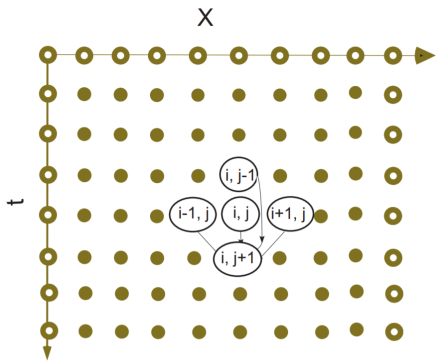


Rysunek: Poszukiwanie rozwiązania $y(x,t)$ dla dyskretnych wartości zmiennych niezależnych x i t na siatce.

- $x = i\Delta x, \quad i = 1, ..., N_x$
- $t = j\Delta t, \quad j = 1, ..., N_t$

$$y(x,t) = y(i\Delta x, i\Delta t) \stackrel{def}{=} y_{i,j}$$

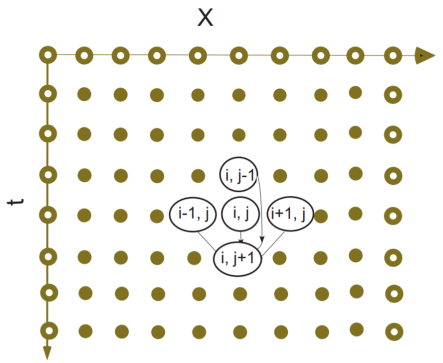
Algorytm kroku czasowego



Rysunek: Poszukiwanie rozwiązania $y(x,t)$ dla dyskretnych wartości zmiennych niezależnych x i t na siatce.

- $\frac{\partial^2 y}{\partial t^2} \simeq \frac{y_{i,j+1}+y_{i,j-1}-2y_{i,j}}{(\Delta t)^2}$, $\frac{\partial^2 y}{\partial x^2} \simeq \frac{y_{i+1,j}+y_{i-1,j}-2y_{i,j}}{(\Delta t)^2}$

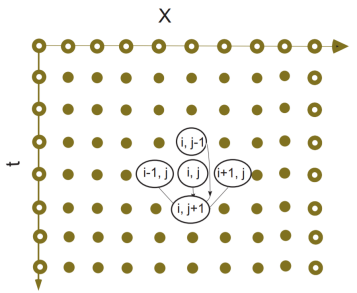
Algorytm kroku czasowego



Rysunek: Poszukiwanie rozwiązania $y(x,t)$ dla dyskretnych wartości zmiennych niezależnych x i t na siatce.

- $\frac{\partial^2 y}{\partial t^2} \simeq \frac{y_{i,j+1}+y_{i,j-1}-2y_{i,j}}{(\Delta t)^2}$, $\quad \frac{\partial^2 y}{\partial x^2} \simeq \frac{y_{i+1,j}+y_{i-1,j}-2y_{i,j}}{(\Delta t)^2}$
- A jeśli wstawimy powyższe do równania falowego $\frac{\partial^2 y(x,t)}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 y(x,t)}{\partial t^2}$, gdzie $c = \sqrt{\frac{T}{\rho}}$, to otrzymamy: $\frac{y_{i,j+1}+y_{i,j-1}-2y_{i,j}}{(c^2\Delta t)^2} = \frac{y_{i+1,j}+y_{i-1,j}-2y_{i,j}}{(\Delta t)^2}$

Algorytm kroku czasowego

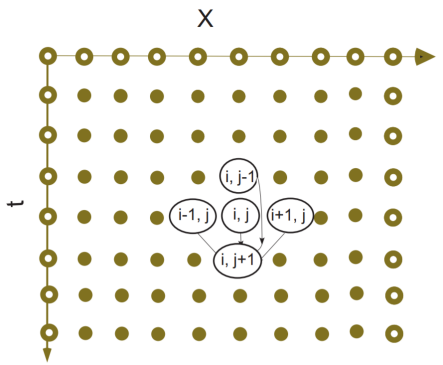


Rysunek: Poszukiwanie rozwiązania $y(x,t)$ dla dyskretnych wartości zmiennych niezależnych x i t na siatce.

- $\frac{\partial^2 y}{\partial t^2} \simeq \frac{y_{i,j+1}+y_{i,j-1}-2y_{i,j}}{(\Delta t)^2}, \quad \frac{\partial^2 y}{\partial x^2} \simeq \frac{y_{i+1,j}+y_{i-1,j}-2y_{i,j}}{(\Delta t)^2}$
- A jeśli wstawimy powyższe do równania falowego $\frac{\partial^2 y(x,t)}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 y(x,t)}{\partial t^2}$, gdzie $c = \sqrt{\frac{T}{\rho}}$, to otrzymamy: $\frac{y_{i,j+1}+y_{i,j-1}-2y_{i,j}}{(c^2\Delta t)^2} = \frac{y_{i+1,j}+y_{i-1,j}-2y_{i,j}}{(\Delta t)^2}$

$$y_{i,j+1} = 2y_{i,j} - y_{i,j-1} + \frac{c^2}{c'^2}[y_{i+1,j} + y_{i-1,j} - 2y_{i,j}], \quad c' \stackrel{def.}{=} \frac{\Delta x}{\Delta t}$$

Algorytm kroku czasowego

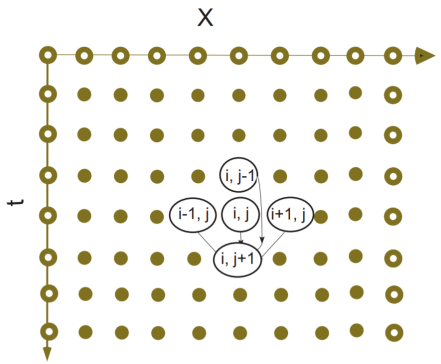


Rysunek: Poszukiwanie rozwiązania $y(x,t)$ dla dyskretnych wartości zmiennych niezależnych x i t na siatce.

$$y_{i,j+1} = 2y_{i,j} - y_{i,j-1} + \frac{c^2}{c'^2}[y_{i+1,j} + y_{i-1,j} - 2y_{i,j}], \quad c' \stackrel{def.}{=} \frac{\Delta x}{\Delta t}$$

- $\frac{\partial y}{\partial t}(x,0) \simeq \frac{y(x,\Delta t) - y(x,-\Delta t)}{2\Delta t} = 0, \quad \Rightarrow \quad y_{i,0} = y_{i,2}$

Algorytm kroku czasowego



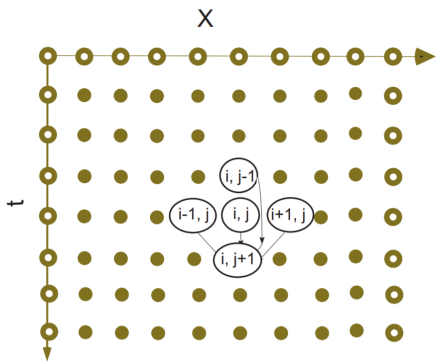
Rysunek: Poszukiwanie rozwiązania $y(x,t)$ dla dyskretnych wartości zmiennych niezależnych x i t na siatce.

$$y_{i,j+1} = 2y_{i,j} - y_{i,j-1} + \frac{c^2}{c'^2}[y_{i+1,j} + y_{i-1,j} - 2y_{i,j}], \quad c' \stackrel{def.}{=} \frac{\Delta x}{\Delta t}$$

- $\frac{\partial y}{\partial t}(x,0) \simeq \frac{y(x,\Delta t) - y(x,-\Delta t)}{2\Delta t} = 0, \quad \Rightarrow \quad y_{i,0} = y_{i,2}$

$$y_{i,2} = y_{i,1} + \frac{c^2}{2c'^2}[y_{i+1,1} + y_{i-1,1} - 2y_{i,1}] \quad \text{dla } j = 2$$

Algorytm kroku czasowego



Rysunek: Poszukiwanie rozwiązania $y(x,t)$ dla dyskretnych wartości zmiennych niezależnych x i t na siatce.

$$y_{i,j+1} = 2y_{i,j} - y_{i,j-1} + \frac{c^2}{c'^2}[y_{i+1,j} + y_{i-1,j} - 2y_{i,j}], \quad c' \stackrel{def.}{=} \frac{\Delta x}{\Delta t}$$

- $\frac{\partial y}{\partial t}(x,0) \simeq \frac{y(x,\Delta t) - y(x,-\Delta t)}{2\Delta t} = 0, \quad \Rightarrow \quad y_{i,0} = y_{i,2}$

$$y_{i,2} = y_{i,1} + \frac{c^2}{2c'^2}[y_{i+1,1} + y_{i-1,1} - 2y_{i,1}] \quad \text{dla } j = 2$$

- $c \leqslant c' = \Delta x/\Delta t$ Warunek Courant'a

Implementacja. Rozwiązanie problemu

```
# EqString.py:  Animated leapfrog solution of wave equation

from visual import *

# Set up curve
g = display(width = 600, height = 300, title = 'Vibrating string')
vibst = curve(x = list(range(0, 100)), color = color.yellow)
ball1 = sphere(pos = (100, 0), color = color.red, radius = 2)
ball2 = sphere(pos = ( - 100, 0), color = color.red, radius = 2)
ball1.pos
ball2.pos
vibst.radius = 1.0

# Parameters
rho   = 0.01                                        # string density
ten   = 40.                                         # string tension
c     = sqrt(ten/rho)                             # Propagation speed
c1    = c                                            # CFL criterium
ratio =  c*c/(c1*c1)

# Initialization
xi = zeros((101,3), float)                          # 101 x's & 3 t's
for i in range(0, 81):      xi[i, 0] = 0.00125*i;                  # IC
for i in range (81, 101):  xi[i, 0] = 0.1 - 0.005*(i - 80)         # IC
for i in range(0, 100):                                       # 1st t step
    vibst.x[i] = 2.0*i - 100.0                           # assign,scale x
    vibst.y[i] = 300.*xi[i, 0]                           # assign, scale y
vibst.pos                                                  # draw string

# Later time steps
for i in range(1, 100): xi[i,1] = xi[i,0] + 0.5*ratio*(xi[i+1,0]+xi[i-1,0]-2*xi[i,0])
while 1:                                  # continue plotting till user quits
    rate(50)                             # delays plotting, (bigger = slower)
    for i in range(1, 100):
        xi[i,2] = 2.*xi[i,1] - xi[i,0] + ratio * (xi[i+1,1]+xi[i-1,1]-2*xi[i, 1])
    for i in range(1, 100):
         vibst.x[i] = 2.*i - 100.0                                 # scaled x
         vibst.y[i] = 300.*xi[i, 2]                                # scaled y
    vibst.pos                                                  # plot string
    for i in range(0, 101):
        xi[i, 0] = xi[i, 1]                                  # recycle array
        xi[i, 1] = xi[i, 2]

print("Done!")
```

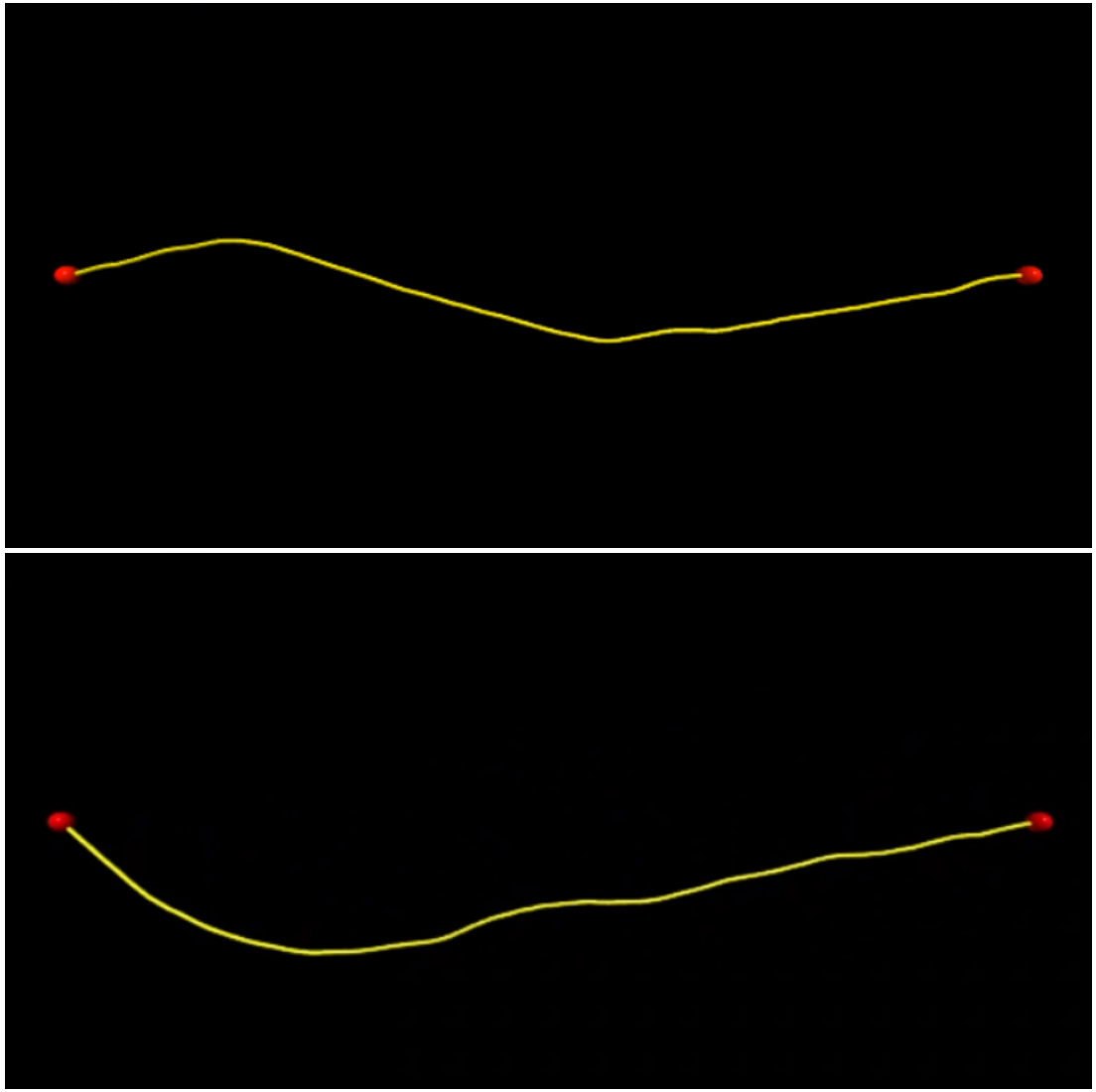
Rysunek: Program wyjściowy stworzony w celu rozwiązywania równania falowego poprzez kroki czasowe dla struny o długości $L = 1m$ z nieruchomymi końcami.

Rozwiązanie problemu

```
c = math.sqrt(ten / rho)  # 4000
c1 = c
steps = 101

g = canvas(width=600, height=300, title="Vibrating string")
string = curve(pos=[vec(i - steps / 2, 0, 0) for i in range(steps)], color=color.yellow, radius=0.5, x0=steps * [0.],
               x1=steps * [0.], x2=steps * [0.])
ball1 = sphere(pos=vector(100, 0, 0), radius=2.0, color=color.red)
ball2 = sphere(pos=vector(-100, 0, 0), radius=2.0, color=color.red)

for i in range(0, 81):
    string.x1[i] = (0.00125 * i + 0.5 * ratio * (0.00125 * (i + 1) + 0.00125 * (i - 1) - 2 * 0.00125 * i))
    string.x0[i] = string.x1[i]
    string.modify(i, y=300 * string.x2[i], x=2. * i - 100.)
for i in range(81, 101):
    string.x1[i] = (0.1 - 0.005 * (i - 80) + 0.5 * ratio * (
                0.1 - 0.005 * (i - 80 + 1) + 0.1 - 0.005 * (i - 80 - 1) - 2 * 0.1 - 0.005
                * (i - 80)))
    string.x0[i] = string.x1[i]
    string.modify(i, y=300 * string.x2[i], x=2. * i - 100.)


nstep = 0
while 1:
    rate(500)

    for i in range(1, 100):
        string.x2[i] = 2. * string.x1[i] - string.x0[i] + ratio * (
                string.x1[i + 1] + string.x1[i - 1] - 2 * string.x1[i])
        string.modify(i, y=300 * string.x2[i], x=2. * i - 100.)

    for i in range(0, 101):
        string.x0[i] = string.x1[i]
        string.x1[i] = string.x2[i]
    nstep += 1
    if nstep == 1: g.waitfor('click')
```

Rysunek: Napisany program rozwiązujący problem drgającej struny.

Implementacja. Podsumowanie



Rysunek: Implementacja kodu rozwiązującego problem drgającej struny.

Bibliografia


Materiały udostępnione w pliku „2Fale.pdf”


https://www.vpython.org/

Dziękujemy za uwagę!