Závěrečný projekt do předmětu Fyzikální akustika ‘MPC-FYA‘

Simulace vlnění na struně kytary s možností volby iniciálního vychýlení, tlumení a dalších parametrů struny.

Bc. Petr Němec[1]

[1]Ústav elektroniky a informačních technologií, Vysoké učení technické v Brně

4. května 2023

Abstrakt

Chování struny hudebního nástroje, jako je například kytara, cimbál, apod. lze velmi snadno popsat pomocí diferenciální vlnové rovnice.

Tento článek se zabývá počítačovou simulací vlnění na struně kytary s důrazem na přechodové jevy, které jsou způsobeny tlumením. Simulace je implementována právě pomocí vlnové rovnice.

Cílem experimentu je zkoumat vliv počátečních podmínek na vlastnosti vlnění. Podmínky mohou být například koeficienty útlumu, délková hustota struny, napětí struny, délka struny, ale také tvar počátečního vychýlení. Počáteční vychýlení struny je nastaveno v čase $t = 0$, následně jsou sledovány změny v čase a poloze.

Výsledky experimentu jsou prezentovány prostřednictvím různých kombinací vstupních parametrů, které ovlivňují vlnění na struně.

Klíčová slova

Vlnová rovnice, Simulace, Struna, Přechodové jevy na struně,

1 Úvod

Tento článek se věnuje počítačové simulaci vlnění na struně kytary. Simulace je navržena tak, aby dokázala brát v úvahu: tvar vychýlení struny v čase $t = 0\,\mathrm{s}$; koeficienty útlumu a jiné „naivně“ pojaté vlastnosti, které mají vliv na tlumení vlnění;

Výchozí hypotézou je vlnová rovnice ve tvaru [1]:

$$\frac{\partial^2 u}{\partial x^2} - \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} + \beta\frac{\partial u}{\partial t} = 0, \tag{1}$$

kde $u(x,t)$ je vychýlení struny závislé na čase t a poloze x, c odpovídá rychlosti vlny a β je koeficient útlumu.

Rychlost c lze z rovnice 1 vyjádřit jako [1]:

$$c = \sqrt{\frac{T}{\rho}}, \tag{2}$$

kde T odpovídá síle napínající strunu na koncích a ρ je délková hustota struny.

Nastavením různých kombinací počátečních hodnot budou porovnány jednotlivé simulace a bude možné všimnout si přechodových jevů.

Tyto jevy souvisí právě s tlumením struny.

2 Materiály a metody

Základem výzkumu je *script*[1] v programovacím jazyku Python, který po spuštění do grafu dvě křivky: jednu, znázorňující počáteční vychýlení struny a druhou animovanou, se simulací jejího průběhu.

Při implementaci bylo využito knihoven:

```
import time
import math
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import
    FuncAnimation
```

Centrální částí kódu je třída `String()`, která implementuje fyzikální model vlny na struně podle vlnové rovnice (1), ze které vyjádříme jednorozměrnou vlnovou rovnici [2]:

[1]Celý script lze prohlédnout např. na: https://github.com/Nemecxpetr/GuitarStringSimulation/blob/main/script/xnemec93.py

$$\frac{\partial^2 u}{\partial t^2} = c^2\left(\frac{\partial^2 u}{\partial x^2} - \beta\frac{\partial u}{\partial t}\right). \quad (3)$$

Vlna se pohybuje na struně, která je „diskretizovaná“ na mřížce s bodovými hodnotami uloženými v poli `y`. Kód obsahuje několik metod, které implementují výpočty pro vlnovou rovnici a tlumení viskozitou. Následující popis vědeckých metod detailněji vysvětlí, jakým způsobem kód funguje.

Kód implementuje numerickou metodu konečných diferencí, která aproximuje derivace vlnové rovnice diskretizací struny na mřížku a výpočtem hodnot v bodech této mřížky.

Metoda `__init__` inicializuje stav struny z globálních proměnných, zejména počáteční hodnoty vychýlení v ose y `y0`, polohy v každém aktuálním krokuy (viz metoda `increment`), rychlosti šíření vlny `c` a „tlumení“ viskozitou `viscosity`.

Metoda `pad_array` přidává "ghost index"na obou koncích struny, čímž jsou ošetřeny okrajové podmínky pro ukotvení struny na obou koncích.

Metoda `increment` provádí s každým zavoláním výpočet dalšího kroku simulace. V první řadě se vypočítá krok času `dt` na základě rychlosti šíření vlny c a vzdálenosti mezi body na mřížce `x`. Poté se vypočítají hodnoty vlny v diskrétních bodech struny pomocí numerické metody pro vlnovou rovnici. Tyto hodnoty se aktualizují pro každý bod v poli `y`, a to jak v závislosti na předchozích hodnotách vlny v aktuálním a předchozím kroku, tak i v závislosti na tlumení viskozitou a tlumení „konstantním“ (koeficient tlumení β).

Animace jako taková je vytvořena pomocí metody `FuncAnimation` z knihovny `matplotlib.pyplot`, která jako vstupní argument potřebuje aktualizační a iniciační funkce a další parametry zajišťující plynulý průběh a zobrazení animace.

Celá simulace byla implementována jako třída `MyAnimation()`

Konstruktor třídy `MyAnimation` se volá při vytváření instance třídy a vytváří instanci objektu `string`, kterou předává do proměnné `self.string`. Dále inicializuje proměnné potřebné pro vizualizaci animace.

Ve funkci `update` jsou vypočteny parametry potřebné pro výpočet animace. Tyto parametry jsou časový úsek a tlumení vlnění. Poté jsou vypočteny nové hodnoty diskretizovaného pole y struny a obraz vlnění se aktualizuje pro danný frame. Nakonec se výsledné hodnoty použijí k aktualizaci grafického prvku "line", který je vykreslován.

Zbývá ještě zmínit, že iniciální tvar vlny je rovněž pole s diskretizovanými hodnotami. Toto pole je naplněno v kódu podmíněným indexováním. Lze si tímto způsobem vymyslet libovolný tvar.

Nejdříve je naplněno pole x od počátečního bodu (1) do bodu délky struny. Poté se inicializuje prázdné pole y podobně jako x.

```
"""
Initial wave shape
"""
# create empty matrix to be filled with
    shape
x = np.linspace(1, 100, 2**8)
y = y = np.empty_like(x)

#v[x, y]
#v[0, 1]

v0 = [.3, 2]
```

Budiž uvedeny dva příklady. První tvar „pily“, který vytvoříme pomocí parametrické rovnice pro přímku a druhý tvat „gaussovi křivky“, která je určena rovněž parametricky:

```

# 'SAW'
y[np.where(x <= v0[0]*x[-1])] =
    v0[1]/(v0[0]*x[-1]) * x[np.where(x
    <= v0[0]*x[-1])]
y[np.where(x > v0[0]*x[-1])] =
    -v0[1]/((1.0-v0[0])*x[-1]) *
    (x[np.where(x > v0[0]*x[-1])] -
    x[-1])

# 'GAUSS'
sigma = 5 # d0 * (x[-1] - x[0]) #
    quality factor the smaller the
    number the steeper the G. function
y = v2[1] * np.exp(-(x -
    v2[0]*x[-1])**2 / (2*sigma**2)) #
    Gaussian function shape
```

Nebyli doposud zmíněny tvary

Celkově tedy tento kód představuje jednoduchou simulaci vlnění na struně kytary, která je následně vizualizována v animaci. Výsledky simulace jsou prezentovány v následujících částech článku.

3 Výsledky simulace

Výsledky experimentu jsou rozděleny podle vstupních parametrů simulovaného systému.

Rychlost vlnění je nastavena na dle rovnice (2).

3.1 Bez tlumení

Jako první se nabízí nastavit simulaci vlnění bez tlumení, tedy:

- $\beta = 0$,

- parametr `viscosity` ve *scriptu* je nastaven na 0,
- délka struny je nastavena na 100 cm,
- $T = 62\,\mathrm{N}$
- $\rho = 6,8\,\mathrm{kg} \cdot \mathrm{m}^3$

V tomto případě se vlna pohybuje z jednoho konce na druhý kde se otočí jeho fáze. Bez tlumení tento děj probíhá do „nekonečna“, resp. do ukončení simulace uživatelem.

3.2 Tlumení pouze koeficientem β

Druhá možnost nastavení je zvýšení koeficientu konstantního útlumu β. na obrázcích je nastaven pomocí parametru `damp`. na hodnotu .

- $\beta = 5 \cdot 10^{-5}$,
- `viscosity` je nastaven na 0,
- délka struny je nastavena na 60 cm,
- $T = 70\,\mathrm{N}$
- $\rho = 6,8\,\mathrm{kg} \cdot \mathrm{m}^3$

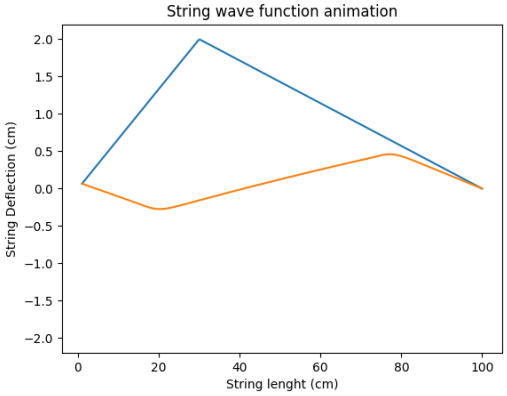
V tomto případě je tvar vlny stejný jako iniciální tvar, ale mění se amplituda jeho vychýlení. Slovem stejný se myslí, takový, jehož „kvalita“ se nemění – jedná-li se např. o Gaussovu křivku pak šířka jejího rozptylu je po celou dobu vlnění stejná. U „pilovitého“ počátečního vychýlení je po celou dobu průběhu vlnění v maximálních hodnotách výchylky charakteristika tvaru struny podobně ostrá „špička“.

Velikost maximálního vychýlení v průběhu vlnění klesá. Kdyby bylo tlumení záporné, pak by rostla, což by ovšem v simulaci způsobilo po krátké chvíli nestabilitu.

3.3 Tlumení pouze parametrem `viscosity`

- $\beta = 0$,
- `viscosity` je nastaven na 400,
- délka struny je nastavena na 100 cm,
- $T = 62\,\mathrm{N}$
- $\rho = 6,8\,\mathrm{kg} \cdot \mathrm{m}^3$

V případě tlumení viskozitou není tak jednoznačně jednoduché popsat co se děje. Oproti konstantnímu tlumení implementovanému v předchozím výsledku se zásadně mění kvalita počátečního tvaru vychýlení struny. To má do jisté míry za důsledek i změnu velikosti amplitudy vychýlení tvaru struny v daném čase.



Obrázek 1: Znázornění tlumení oběma parametry

3.4 Tlumení pomocí obou parametrů

- $\beta = 10 \cdot 10^{-5}$,
- `viscosity` je nastaven na 1200,
- délka struny je nastavena na 100 cm,
- $T = 75\,\mathrm{N}$
- $\rho = 6,8\,\mathrm{kg} \cdot \mathrm{m}^3$

Nastavením obou parametrů dochází ke kombinaci předchozích jevů.

4 Diskuze

Pokud není přítomné tlumení pohybu, pohybuje se vlna stále stejným způsobem a nedochází ke vzniku přechodových jevů. Jelikož jedna z počátečních podmínek je, uchycení struny na obou koncích dochází k periodickému otáčení fáze vlnění.

Mimo jiné lze pozorovat, že při změně parametru velikosti délky struny se mění rychlost průběhu animace, která odpovídá frekvenci vlnění f. To odpovídá předpokladu, že frekvence f je přímo úměrná rychlosti šíření vlny, která je v simulaci konstantní definovaná parametry struny podle rovnice (2) a nepřímo úměrná vlnové délce, která v tomto případě odpovídá délce struny:

$$f = \frac{v}{\lambda} \tag{4}$$

Obdobně dochází ke zrychlení animace změnou parametrů T, nebo c. Vzhledem ke zvolené metodě je ovšem potřeba volit tyto parametry opatrně. Zvolená aproximační metoda počítá za pomoci konečného časového přírůstku `dt`. Ten může být pro příliš velké rychlosti velký a systém

přestane být stabilní. Pro potřebu větších rychlostí je možné parametr dt zmenšit, čímž se ale zvýší náročnost výpočtu.

Pokud se v systému již nachází tlumení, pak lze pozorovat přechodové jevy. Ty se projevují změnou tvaru a maximální výchylky tvaru vlny. V případě že použijeme model s „viskózním“ tlumením. Lze pozorovat, že jsou nejdříve tlumeny vyšší frekvence kmitání vlastních částic struny.

Pro potřebu hlubšího porozumění lze zvolit jako počáteční tvar vlnu složenou ze součtu několika funkcí sinus, jejich frekvence jsou násobky.

V našem kódu např. tímto způsobem:

```
# 'SINE' - for visualising transients
f1, f2, f3, f4= .5, 2, 4, 8
y[1:-1] =
    v0[1]*np.sin(f1*2*np.pi/(x[-1] -
    x[0]) * x[1:-1])
y[1:-1] += v0[1]*np.sin(f2*np.pi/(x[-1]
    - x[0]) * x[1:-1])
y[1:-1] += v0[1]*np.sin(f3*np.pi/(x[-1]
    - x[0]) * x[1:-1])
y[1:-1] +=
    v0[1]*np.sin(f4*2*np.pi/(x[-1] -
    x[0]) * x[1:-1])
```

Průběh s tlumením zobrazen na obr. 2. Modře je vyznačen výchozí tvar vlny. V tomto případě si jej lze představit rovněž jako tvar spektra kmitání složeného ze stejných funkcí sinus, jaké byli použity pro tuto simulaci. Oranžově je potom vyznačen průběh v čase n. Odtud je vidět, že z vlny postupně vymizely složky s vyššími frekvencemi.

5 Závěr

Simulování vln na struně a animace tohoto chování je pomocí navrženého *scriptu* plynulé a funkční, pokud jsou dodrženy podmínky pro stabilitu systému.

Simulace vychází z numerické diskretizace vlnové rovnice pro jednorozměrnou vlnu. Změnou vstupních parametrů do systému byli provedeny různé simulace.

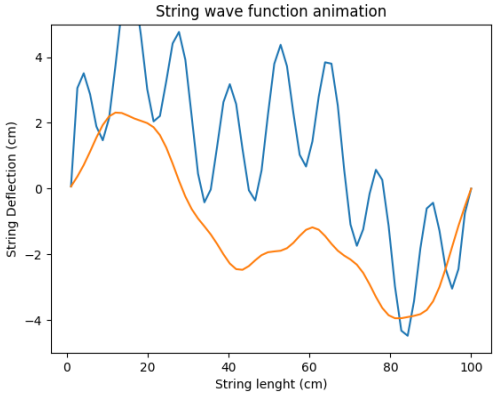
Experimenty byly provedeny bez tlumení, s konstantním tlumením a s tlumením závislým na rychlosti.

Výsledky experimentů ukazují, že počáteční podmínky mají vliv na vlnění na struně a že různé kombinace vstupních parametrů vedou k různým přechodovým jevům.

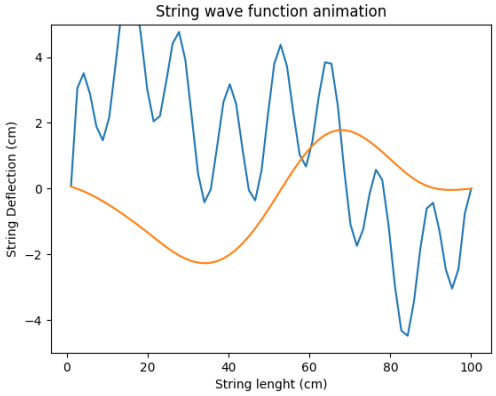
Budoucí směry výzkumu by mohly zahrnovat další experimenty s různými materiály strun, vylepšení numerických metod použitých v simulaci. Rozšíření o další dimenze a simulace vlnění například na rovinné bláně, apod. ...

6 Přílohy

6.1 Obrázky ze simulací



Obrázek 2: Znázornění přechodových jevů – 1/3

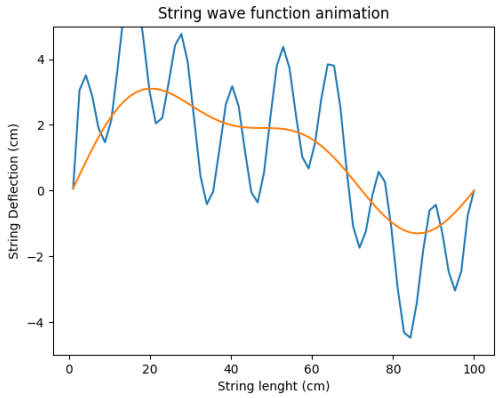


Obrázek 3: Znázornění přechodových jevů – 2/3

6.2 Vybrané části kódu

Třída `String()` implementující vlnovou rovnici:

```
class String():
    def __init__(self, x, y0, c):
        # Initialize the string's state
        self.y = self.pad_array(y0)
        self.x = self.pad_array(x)
        self.y0 = self.pad_array(y0)
        self.c = c
        self.viscosity = viscosity
        self.y_prev = np.copy(self.y0)
    # Pad the array with ghost points
    to avoid boundary conditions
    def pad_array(self, arr):
        padded_arr =
    np.concatenate(([arr[0]] *
    (len(arr)-1), arr, [arr[-1]] *
    (len(arr)-1)))
        return padded_arr[len(arr)-1 :
    -len(arr)+1]
    def increment(self, dt):
        """Increment shape of string by
    dt with viscosity damping"""
```



Obrázek 4: Znázornění přechodových jevů – 3/3

```
        # Calculate the time step based on the string speed and grid spacing
        r = (self.c * dt / np.gradient(self.x))**2
        # Update the string shape using the wave equation with viscosity damping
        temp = np.copy(self.y)
        self.y[1:-1] = 2 * self.y[1:-1] - self.y_prev[1:-1] + r[1:-1] * (self.y[2:] - 2 * self.y[1:-1] + self.y[:-2])
        self.y[1:-1] += self.viscosity * dt * r[1:-1] * (self.y[2:] + self.y[:-2] - 2 * self.y[1:-1])
        self.y[1:-1] -= damp * (self.y[1:-1] - self.y_prev[1:-1]) / dt
        # Update the previous string shape for the next time step
        self.y_prev = temp
```

Funkce `update` z třídy `MyAnimation` implementující animaci:

```
    def update(self, frame_no):
        global string
        # Real-time plotting parameters
        t0 = time.time()
        t = t0
        t_next = time.time() + interval
        # while the time of the simulation has started
        while t_next <= t0:
            # Increment the string shape by the time step and damping factor
            string.increment(dt)
            t+=dt
        # Increment the string shape for the current frame with a damping factor
        string.increment(dt)
        line.set_ydata(string.y)
        return line,
```

Reference

[1] Langer J, Podolský J. Teoretická mechanika; 2013. Studijní text k přednášce NOFY003 - „Teoretická mechanika“. Available from: http://utf.mff.cuni.cz/vyuka/AFY016/TEXTY/kontinuum.pdf.

[2] prof RNDr Július Krempaský D, prof Ing František Schauer D. Fyzika I a II; 2004. Univerzita Tomáše Bati ve ZlíněSlovenská technická univerzita v Bratislavě. Available from: http://kf-lin.elf.stuba.sk/KrempaskyFyzika/.