3. Simulácie fyzikálnych pohybov

Kľúčové slová

simulácia, fyzikálne pohyby, trajektória, animácia, analytický model, diferenčný model, grafické rozhranie, generátor pseudonáhodných čísel

Čo sa naučíme a čo si precvičíme

- Analyzovať vybrané fyzikálne javy a zostaviť pre nich analytický model (tiež statický model), resp. diferenčný model (tiež dynamický model) pohybujúceho sa bodu podľa fyzikálnych zákonov (Newtonových pohybových zákonov, Coulombovho zákona),
- naprogramovať simuláciu pohybujúceho sa bodu využívajúcu grafické rozhranie s knižnicou Tkinter s viacerými ovládacími prvkami (Label, Entry, Button, Canvas, OptionMenu),
- použiť simulácie pohybujúceho sa bodu na overenie, resp. vyvrátenie hypotéz,
- aplikovať simulácie pohybujúceho sa bodu pri programovaní počítačových hier,
- pri tvorbe programov použiť (precvičiť) príkazy opakovania, vetvenia a priradenia, tvorbu vlastných funkcií, príkazy karteziánskej grafiky (`tkinter`), príkazy modulov `math` a `random`.

3.1 Rovnomerný priamočiary pohyb

Výkladový text:

Podľa 1. Newtonovho pohybového zákona platí, že teleso zotrváva v pokoji alebo v rovnomernom priamočiarom pohybe, kým nie je nútené vonkajšími silami tento svoj stav zmeniť. Pre zjednodušenie simulácií pohybov telesa budeme ďalej zanedbávať rozmery telesa a budeme uvažovať o tzv. **hmotnom bode** (ďalej len bode). **Trajektóriou** rovnomerného priamočiareho pohybu bodu je časť priamky. Pre **dráhu** s rovnomerného priamočiareho pohybu bodu platí vzťah: $s = vt$, kde v je rýchlosť rovnomerného priamočiareho pohybu bodu a t je doba trvania rovnomerného priamočiareho pohybu.

Ak sa bod pohybuje rovnomerným priamočiarym pohybom rýchlosťou $v = (v_x, v_y)$ a v čase $t = 0$ sú jeho súradnice polohy rovné $(0, 0)$, tak pre jeho súradnice polohy x, y v ľubovoľnom čase *t* platí vzťah:

$$(x, y) = (v_x, v_y)t$$

alebo pre každú súradnicu polohy:

$$x = v_x t \qquad y = v_y t$$

Úloha 1

a) Analyzujte a so spolužiakom prediskutujte, čo robí uvedený program `rovnomerny1.py`.

```
# rovnomerny1.py
import tkinter

okno = tkinter.Tk()
platno = tkinter.Canvas(okno, width=300, height=300)
platno.grid()
v_x, v_y = 10, 15
for t in range(0, 21):
    x = v_x * t
    y = v_y * t
    platno.create_oval(x - 3, y - 3, x + 3, y + 3, fill='yellow')

okno.mainloop()
```

b) Upravte program tak, aby zobrazoval trajektóriu pohybu bodu v súradnicovom systéme s osou y nasmerovanou nahor, pričom veľkosť zobrazovaného bodu je väčšia ako predtým (napr. 10) a zobrazená inou farbou (napr. azúrovou, angl. cyan). Výsledný program uložte do súboru `rovnomerny1_R.py`.

Úloha 2

Analyzujte a so spolužiakom prediskutujte, čo robí uvedený program `rovnomerny2.py`.

```
# rovnomerny2.py
import tkinter

okno = tkinter.Tk()
platno = tkinter.Canvas(okno, width=300, height=300)
platno.grid()
x, y = 0, 0
v_x, v_y = 2, 3
bod = platno.create_oval(x - 2, y - 2, x + 2, y + 2, fill='red')
for t in range(0, 100):
    x = v_x * t
    y = v_y * t
    y = 300 - y
    platno.coords(bod, x - 2, y - 2, x + 2, y + 2)
    platno.update()
    platno.after(50)

okno.mainloop()
```

a) **V čom sa líši** uvedený program od programu `rovnomerny1.py` z úlohy 1?
b) Aký význam majú metódy **platno.coords(), platno.update()** a **platno.after()**?
c) Aký význam má pred cyklom uvedené priradenie **bod = platno.create_oval()**?

Výkladový text:

Ak chceme na plátne zanechať trajektóriu pohybujúceho sa bodu, stačí opakovane použiť metódu **platno.create_oval()**, ktorá na plátne vytvorí súbor kruhov reprezentujúcich trajektóriu pohybujúceho sa bodu.

Ak chceme animovať pohyb bodu pomocou jedného kruhu, vytvoríme objekt **bod** pomocou priradenia **bod = platno.create_oval()**, ktorého súradnice polohy na plátne meníme metódou **platno.coords()**. Aby sme nevideli len poslednú pozíciu pohybujúceho sa bodu potrebujeme občerstviť obrazovku pri každej zmene súradníc bodu, čo dosiahneme pomocou metódy **platno.update()**. Na časové zdržanie (v milisekundách) použijeme metódu **platno.after()**.

Úloha 3

Vytvorte program `rovnomerny3_R.py` vykresľujúci trajektóriu rovnomerného priamočiareho pohybu bodu, ktorého grafické rozhranie umožňuje zadanie súradníc vektora rýchlosti (v_x, v_y), zmazanie plátna a ukončenie behu programu. (Nápoveď: Na zadanie hodnôt rýchlosti, vykreslenie trajektórie, zmazanie plátna a ukončenie programu použite prvky **Entry**, **Label**, **Button**, **Canvas** a riadiace premenné Tkinteru typu **DoubleVar** a metódu **get()** na ďalšie použitie v programe hodnoty zadanej v prvku **Entry**.)

Výkladový text:

Pri tvorbe programov s väčším počtom prvkov grafického rozhrania sa stáva programový kód menej prehľadným. Alternatívou tvorby programu s grafickým rozhraním je použitie objektového prístupu (OOP), ktorý umožňuje napísať kompaktnejší programový kód. V alternatívnom OOP riešení úlohy 3, sme vytvorili triedu **Simulacia** odvodenú od triedy **Tk**, v ktorej sme definovali tri funkcie tejto triedy tzv. **metódy**. V metóde **__init__** nastavujeme počiatočné vlastnosti a spúšťame ďalšie metódy. V metóde **vytvorGUI** vytvárame grafické rozhranie programu a v metóde **kresli()** vykresľujeme na plátno trajektóriu pohybujúceho sa bodu. Metódy a premenné s prefixom **self** patria triede **Simulacia** a sú prístupné v každej jej metóde. Objekt triedy **Simulacia** vytvárame a spúšťame zápisom **Simulacia()**.

```
# rovnomerny4.py
import tkinter

class Simulacia(tkinter.Tk):
    def __init__(self):
        '''Nastavenie počiatočných vlastnosti a spúšťanie ďalších metód triedy'''
        super().__init__()
        self.title('Rovnomerný priamočiary pohyb')
        self.vytvorGUI()
        self.mainloop()

    def vytvorGUI(self):
        '''Vytvorenie grafického rozhrania programu (GUI)'''
        tkinter.Label(self, text='v_x =').grid(row=0, column=0)

        self.v_x = tkinter.DoubleVar()
        self.v_x.set(10)
        tkinter.Entry(self, textvariable=self.v_x).grid(row=0, column=1)
```

```
        tkinter.Label(self, text='v_y =').grid(row=0, column=2)

        self.v_y = tkinter.DoubleVar()
        self.v_y.set(15)
        tkinter.Entry(self, textvariable=self.v_y).grid(row=0, column=3)

        tkinter.Button(self, text='Kresli', command=self.kresli).grid(row=1,
column=0)
        tkinter.Button(self, text='Ukonči', command=self.destroy).grid(row=1,
column=3)
        self.platno = tkinter.Canvas(self, width=400, height=400,
background='#FFF')
        self.platno.grid(row=2, column=0, columnspan=4)

    def kresli(self):
        '''Vykreslenie trajektórie rovnomerného priamočiareho pohybu bodu'''
        v_x = self.v_x.get()
        v_y = self.v_y.get()
        for t in range(0,21):
            x = v_x * t
            y = v_y * t
            y = 400 - y
            self.platno.create_oval(x - 2, y - 2, x + 2, y + 2, fill='yellow')

Simulacia()
```

Úloha 4

Upravte programový kód súboru `rovnomerny4.py` tak, aby program vykreslil väčšie plátno a k nemu odpovedajúci počet bodov trajektórie. Do grafického rozhrania doplňte textové pole na zadanie veľkosti kruhu reprezentujúceho hmotný bod. Výsledný program uložte do súboru `rovnomerny4_R.py`. (Nápoveď: Výšku plátna reprezentuje výraz **int(self.platno['height'])**, resp. výraz **float(self.platno['height'])**.)

3.2. Rovnomerne zrýchlený pohyb

Výkladový text:

Pozrime sa teraz na najjednoduchší prípad nerovnomerného pohybu, a to **rovnomerne zrýchleného priamočiareho pohybu**. Pre dráhu s rovnomerne zrýchleného pohybu bodu v ľubovoľnom čase t, ktorý sa pohybuje s konštantným zrýchlením a a v čase $t = 0$ mal počiatočnú rýchlosť v_0 a prejdenú počiatočnú dráhu s_0, platí vzťah:

$$s = s_0 + v_0 t + \frac{1}{2} a t^2$$

Vo všeobecnosti, ak uvažujeme rovnomerne zrýchlený pohyb bodu v rovine s vektorom stáleho zrýchlenia $a = (a_x, a_y)$, vektorom počiatočnej rýchlosti $v_0 = (v_{0x}, v_{0y})$ a počiatočnou polohou (x_0, y_0) v čase $t = 0$, môžeme podľa princípu nezávislosti pohybov uvažovať o tomto pohybe ako o zložení dvoch pohybov, napr. v smere súradnicových osí x a y. Potom pre súradnice polohy bodu x a y v ľubovoľnom čase t platia vzťahy:

$$x = x_0 + v_{0x} t + \frac{1}{2} a_x t^2$$

$$y = y_0 + v_{0y}t + \frac{1}{2}a_y t^2$$

kde x_0 a y_0 sú počiatočné súradnice polohy bodu, v_{0x} a v_{0y} sú súradnice vektora počiatočnej rýchlosti, a_x a a_y sú súradnice vektora konštantného zrýchlenia.

Úloha 5

a) Vytvorte program `zrychleny1_R.py` vykresľujúci trajektóriu bodu pohybujúceho sa rovnomerne zrýchlene, ktorý umožňuje zadanie súradníc počiatočnej polohy bodu (x_0, y_0), vektora zrýchlenia (a_x, a_y), vektora počiatočnej rýchlosti (v_{0x}, v_{0y}), celkového času pohybu $t_{celkový}$, zmazanie plátna a ukončenie behu programu.

b) Preskúmajte nastavením ktorých hodnôt dosiahnete vykreslenie trajektórie rovnomerne zrýchleného pohybu (napr. rovnomerne zrýchleného priamočiareho pohybu, vodorovného či šikmého vrhu).

3.3. Pohyb bodu, na ktorý pôsobia viaceré sily

Výkladový text:

V predchádzajúcich simuláciách rovnomerného priamočiareho a rovnomerne zrýchleného pohybu bodu sme použili tzv. **analytický model,** resp. **analytické vyjadrenie súradníc** pohybujúceho sa bodu.

Pri simuláciách pohybu bodu, na ktorý pôsobia viaceré sily, môže byť analytické vyjadrenie súradníc veľmi náročné. Preto sa v praxi používa aj tzv. **diferenčný model**, resp. **diferenčné vyjadrenie súradníc** pohybujúceho sa bodu. Pri **diferenčnom modelovaní** pohybu bodu budeme uvažovať, že v každom patrične malom časovom intervale Δt sa bod sa pohybuje rovnomerným priamočiarym pohybom a ďalšie zmeny pohybového stavu môžu nastať až v nasledovnom časovom intervale Δt. Toto zjednodušenie umožňuje veľmi ľahko simulovať pohyb bodu, na ktorý pôsobia viaceré sily, no na druhej strane ide len o približné výpočty. Čím menší je časový interval Δt, tým presnejšie je modelovanie a bližšie k skutočným výpočtom. Pri diferenčnom modelovaní budeme postupne v rámci každého časového intervalu Δt meniť súradnice vektora rýchlosti v závislosti od aktuálnych súradníc vektora zrýchlenia (udeleného výslednicou síl) a následne meniť súradnice polohy pohybujúceho sa bodu v závislosti od aktuálnych súradníc vektora rýchlosti, čo vyjadríme pomocou nasledovných vzťahov:

$$(v_x, v_y) \leftarrow (v_x, v_y) + (a_x, a_y)\Delta t$$

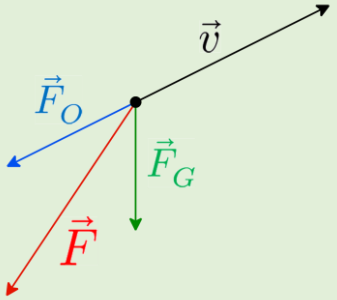
$$(x, y) \leftarrow (x, y) + (v_x, v_y)\Delta t$$

Uvedené vzťahy použijeme pri niekoľkých nasledujúcich úlohách na vykreslenie trajektórie pohybujúceho bodu, na ktorý pôsobia viaceré sily.

3.3.1. Balistická krivka

Výkladový text:

Trajektóriou šikmého vrhu bodu v hmotnom prostredí je **balistická krivka.** Na bod pohybujúci sa rýchlosťou v pôsobia dve sily: gravitačná sila F_G a odporová sila F_O.



*Obr. 4 Výsledná sila **F** pôsobiaca na pohybujúci sa bod ako vektorový súčet odporovej **F$_O$** a gravitačnej sily **F$_G$***

Pri výpočte výsledného zrýchlenia, ktoré udeľuje bodu pôsobiaca výsledná sila, budeme vychádzať z 2. Newtonovho zákona, ktorý je reprezentovaný vzťahom: $a = \frac{F}{m}$.

Gravitačná sila F_G udeľuje pohybujúcemu sa bodu zrýchleníe $a = (0, -g)$, kde v našich zemepisných šírkach je $g \cong 9{,}81\ \mathrm{m.s^{-2}}$. Odporová sila $F_O = -kv$ udeľuje bodu pohybujúceho sa rýchlosťou v zrýchleníe $a_O = (-\frac{k}{m}v_x, -\frac{k}{m}v_y)$.

Výslednica všetkých síl pôsobiacich na pohybujúci sa bod $F = F_G + F_O$ mu udeľuje výsledné zrýchlenie $a = (-\frac{k}{m}v_x, -\frac{k}{m}v_y - g)$.

Úloha 6

a) Vytvorte program `balistika1_R.py` vykresľujúci trajektóriu bodu vystreleného počiatočnou rýchlosťou v_0 pod uhlom α pohybujúceho sa v gravitačnom poli Zeme v hmotnom prostredí (napr. vzduchu) so silou odporu prostredia $F_O = -kv$. Predpokladáme, že počiatočné súradnice polohy bodu (x_0, y_0) majú hodnotu $(0, 0)$ a hmotnosť bodu $m = 1\ \mathrm{kg}$. Program má umožniť zadanie hodnoty počiatočnej rýchlosti v_0, uhla šikmého vrhu α, koeficientu trenia k, časového intervalu Δt, výpis dĺžky doletu x_{max} a času doletu t_{max}, zmazanie plátna a ukončenie behu programu.

b) Po vytvorení programu experimentujte s rôznymi hodnotami časového intervalu Δt, napr. $\Delta t \in \{1; 0{,}1; 0{,}01; 0{,}001; 0{,}0001\}$ a zisťujte presnosť výpočtu dĺžky doletu a času doletu šikmého vrhu bez odporu prostredia. Tieto hodnoty porovnajte s teoretickými hodnotami $x_{max} = \frac{{v_0}^2 \sin 2\alpha}{g}$ a $t_{max} = \frac{2v_0 \sin\alpha}{g}$.

c) Pri akom uhle je dĺžka doletu vo vákuu maximálna? Ako je to s maximálnym doletom v hmotnom prostredí, platí rovnaký uhol ako vo vákuu?

d) Experimentujte s hodnotami koeficientu trenia k, napr. $k \in \{0{,}0; 0{,}1; 0{,}2; 0{,}3; 0{,}4; 0{,}5\}$ a zistite ako sa mení dĺžka doletu x_{max} v závislosti od zmeny koeficientu trenia k.

(Nápoveď: Pri výpočte hodnôt súradníc vektora rýchlosti (v_x, v_y) zo zadanej počiatočnej rýchlosti v_0 a zadaného uhla $\propto$ môžeme použiť vzťah $(v_x, v_y) = (v_0 \cos\alpha\,, v_0 \sin\alpha)$. Pri ich

programovaní môžeme použiť modul **math** a jeho funkcie **math.cos()**, **math.sin()**, **math.radians()**.

Pri programovaní zadávania časového intervalu Δt zo zadanej množiny hodnôt môžeme použiť ovládací prvok výberový zoznam **OptionMenu** s uvedeným zdrojovým kódom:

```
tkinter.OptionMenu(self, self.dt, '0.0001', '0.001', '0.01', '0.1', '1.0')
```

Pri programovaní výpisu hodnôt odporúčame použiť f-reťazce.)

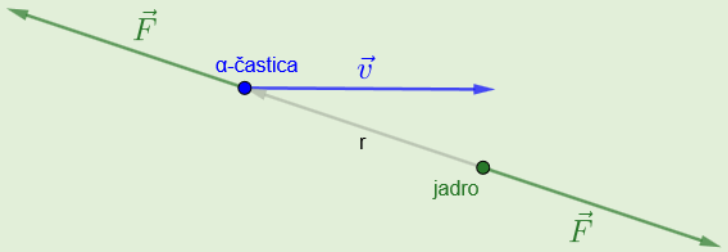
3.3.2. Rutherfordov experiment

Historická poznámka:

V roku 1909 skúmali na návrh Ernesta Rutherforda jeho asistenti Hans Geiger a Ernest Marsden prechod častíc (katiónov ${}^{4}_{2}He^{2+}$) zlatou fóliou. Pozorovali, že väčšina častíc prechádza fóliou. Neskôr zistili, že menší počet častíc sa veľmi odchyľuje od pôvodného smeru pohybu častíc alebo sa dokonca odráža pred fóliu, čo bolo v rozpore s **Thomsonovým modelom atómu**, podľa ktorého mala byť kladný náboj atómu rovnomerne rozmiestnený v celom objeme atómu. Po tomto experimente bol prijatý nový tzv. **Rutherfordov model atómu**, podľa ktorého, je kladný náboj atómu sústredený v jeho malej časti, v tzv. jadre atómu.

Výkladový text:

Poďme vytvoriť program, ktorý bude simulovať **Rutherfordov experiment**, t.j. pohyb kladných α-častíc v blízkosti kladné nabitého jadra atómu zlata. Na túto simuláciu použijeme diferenčný model. V našom prípade uvažujeme o elektrostatickej sile pôsobiacej medzi dvomi bodovými nábojmi – kladnou α-časticou a kladným jadrom zlata. Sila, ktorou pôsobí α-častica na jadro zlata je rovnako veľká ako sila, ktorou pôsobí jadro zlata na α-časticu. Vzhľadom na oveľa väčšiu hmotnosť jadra ako α-častice, budeme pre zjednodušenie predpokladať, že jadro sa prakticky nebude pohybovať a ďalej sa budeme zaoberať len simuláciou pohybu α-častice.



*Obr. 6 Na pohybujúcu sa α-časticu pôsobí jadro elektrostatickou silou **F** v smere polohového vektora od jadra atómu k α-častici*

Pre elektrostatickú silu medzi dvomi bodovými nábojmi q_1 a q_2 vo vákuu so vzdialenosťou r medzi sebou platí Coulombov zákon $F = \frac{1}{4\pi\varepsilon_0}\frac{q_1 q_2}{r^3}r$. Využitím 2. Newtonovho zákona dostaneme vzťah $ma = \frac{1}{4\pi\varepsilon_0}\frac{q_1 q_2}{r^3}r$, z ktorého vyjadríme zrýchlenie $a = \frac{kr}{r^3}$, kde $k = \frac{q_1 q_2}{4\pi\varepsilon_0 m}$ je výraz tvorený veličinami, ktoré sa nebudú meniť počas simulácie. Vektor $r = (r_x, r_y)$ je určený počiatočným bodom – jadrom zlata (x_0, y_0) a koncovým bodom – pohybujúcou sa α-časticou (x, y), t. j. $r = (r_x, r_y) = (x - x_0, y - y_0)$. Pre aktuálne súradnice vektora zrýchlenia platí:

$$(a_x, a_y) = \left(\frac{kr_x}{r^3}, \frac{kr_y}{r^3}\right) = (x - x_0, y - y_0)\frac{k}{r^3}, \text{kde } r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$$

Pri diferečnom modelovaní pohybu α-častíc v blízkosti jadra atómu zlata využijeme aktuálne súradnice vektora zrýchlenia na výpočet aktuálnych súradníc vektora rýchlosti a tie následne na výpočet aktuálnych súradníc polohy pohybujúcej sa α-častice podľa už známych vzťahov (uvedených na strane 9):

$$(v_x, v_y) \leftarrow (v_x, v_y) + (a_x, a_y)\Delta t$$

$$(x, y) \leftarrow (x, y) + (v_x, v_y)\Delta t$$

Zdrojový kód metódy **kresli()** vykresľujúcej zadaný počet letiacich α-častíc so zadaným časovým intervalom a vypisujúcej počet odrazených α-častíc môže vyzerať nasledovne:

```
def kresli(self):
    '''Vykreslenie trajektórie alfa-častíc letiacich okolo jadra zlata'''
    pocet = self.pocet.get()     # počet lúčov
    dt = self.dt.get()           # časový interval
    k = 300                      # výraz q1 * q2 / (4 * pi * epsilon0 * m)
    x0, y0 = 200, 300            # počiatočné súradnice polohy jadra
    odrazene = 0                 # počet odrazených alfa častíc
    self.platno.create_oval(x0 - 9, y0 - 9, x0 + 9, y0 + 9, fill='gold')
    for i in range(pocet):
        x, y = 1, random.randrange(1, 600)
        v_x, v_y = 5, 0
        while 0 < x < 800 and 0 < y < 600:
            koeficient = k / (((x - x0) ** 2 + (y - y0) ** 2) ** (3/2))
            a_x = (x - x0) * koeficient  # aktuálne súradnice zrýchlenia
            a_y = (y - y0) * koeficient
            v_x = v_x + a_x * dt         # aktuálne súradnice rýchlosti
            v_y = v_y + a_y * dt
            x = x + v_x * dt             # aktuálne súradnice polohy
            y = y + v_y * dt
            self.platno.create_oval(x - 1, y - 1, x + 1, y + 1, outline = 'blue')
            self.platno.update()
            if x < 1:
                odrazene +=1
    self.vypis.set(f'{odrazene} odrazených alfa častíc')
```

Kompletný zdrojový kód programu je uložený v súbore `rutherford.py`.

Úloha 7

Po vytvorení programu `rutherford.py` simulujte Rutherfordov experiment pre rôzne hodnoty parametrov. Pozorujte a prediskutujte so spolužiakmi nasledovné otázky:

- Aký počet odrazených α-častíc od jadra atómu zlata ste zaregistrovali?
- Aké rôzne typy trajektórií α-častíc ste zaregistrovali?
- Čo tvorí množinu všetkých trajektórií α-častíc?

3.3.3. Lissajousove krivky

Výkladový text:

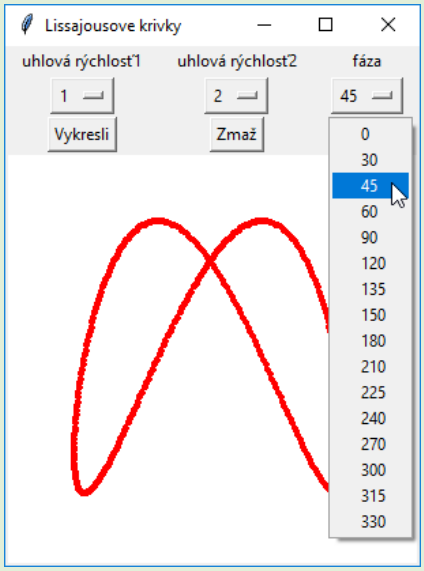
Lissajousove krivky (pomenované podľa francúzskeho fyzika Julesa Antoine Lissajousa) popisujú trajektóriu bodu pohybujúceho sa zloženým pohybom, ktorý je tvorený dvomi na seba kolmými harmonickými pohybmi. Trajektória takého pohybu je popísaná rovnicami:

$$x(t) = a_1 \sin \omega_1 t \qquad\qquad y(t) = a_2 \sin \omega_2 t + \varphi$$

kde a_1 je amplitúda, ω_1 je uhlová rýchlosť 1. harmonického pohybu a a_2 je amplitúda, ω_2 je uhlová rýchlosť a φ je fázový posun 2. harmonického pohybu vzhľadom k 1. harmonickému pohybu.

Pre lepšiu predstavu vykreslenia Lissajousových kriviek sa môžeme pozrieť na video s ich mechanickou simuláciou https://www.youtube.com/watch?v=4CbPksEl51Q.

Poďme naprogramovať simuláciu vykresľovania Lissajousových kriviek. Grafické rozhranie programu môže obsahovať tlačidlá a rozbaľovacie ponuky pre parametre ω_1, ω_2 a φ:



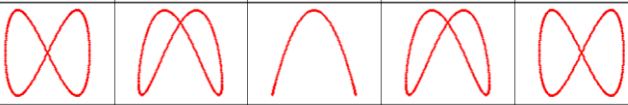
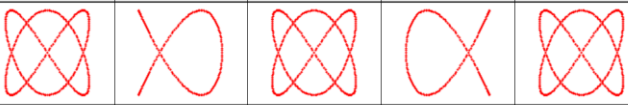
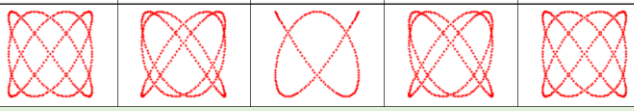
Obr. 8 Grafické rozhranie programu simulujúceho vykresľovanie Lissajousových kriviek

Zdrojový kód metódy **kresli()** vykresľujúci Lissajousove krivky pre rôzne uhlové rýchlosti ω_1 a ω_2 a rôznu fázu φ môže vyzerať nasledovne:

```
def kresli(self):
    '''Vykreslenie Lissajousovej krivky'''
    o1 = self.o1.get()
    o2 = self.o2.get()
    fi = self.fi.get()
    for t in range(360):
        x =  100 * math.sin(math.radians(t * o1 + 0))  + 150
        y = -100 * math.sin(math.radians(t * o2 + fi)) + 150
        self.platno.create_oval(x - 1, y - 1, x + 1, y + 1, outline='red',
fill='red')
```

Kompletný zdrojový kód programu je uložený v súbore `lissajouse.py`.

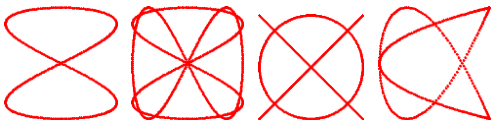
Nastavením rôznych hodnôt uhlových rýchlostí ω_1 a ω_2 a fázy φ dostaneme veľmi zaujímavé obrázky, ako napr.:

ϕ / $\omega_1 : \omega_2$	0°	45°	90°	135°	180°
1 : 1					
1 : 2					
2 : 3					
3 : 4					

Obr. 9 Lissajousove krivky pre rôzne pomery uhlových rýchlosti a rôzne fázy

Úloha 8

Spustite program `lissajouse.py` a nastavením parametrov vykreslite rôzne obrázky, napr.:



3.3.4. Ďalšie zaujímavé krivky

Napokon si ukážeme ďalšie zaujímavé krivky, ktoré by sme mohli ľahko naprogramovať. Pri kotúľaní kružnice s polomerom r zvonku po inej kružnici s polomerom R trajektória jej pevného bodu vytvára krivku – **epicykloidu**, ktorej parametrické rovnice sú:

$$x\ \theta\ =\ R + r\ \cos\theta - r\cos\left(\frac{R+r}{r}\theta\right)$$

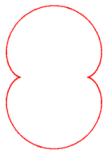
$$y\ \theta\ =\ R + r\ \sin\theta - r\sin\left(\frac{R+r}{r}\theta\right)$$

Ak označíme $k = \frac{R}{r}$ dostaneme nasledovné rovnice:

$$x\ \theta\ = r\ k+1\ \cos\theta - r\cos\ (k+1)\theta$$

$$y\ \theta\ = r\ k+1\ \sin\theta - r\sin\ (k+1)\theta$$

Ak je pomer polomerov k racionálnym číslom, epicykloida je uzavretá.

				
k = 1 (kardioida)	k = 2 (nefroida)	k = 3	k = 5 / 3	k = 19 / 5

Obr. 10 Ukážky epicykloid pre vybrané hodnoty parametra k

Medzi ďalšie zaujímavé krivky patria:

- **hypocykloidy** – krivky tvorené trajektóriou pevného bodu kružnice, ktorá sa kotúľa vnútri inej kružnice bez kĺzania,
- **hypotrochoidy** – krivky tvorené trajektóriou pevného bodu ležiaceho na predĺženom polomere kružnice, ktorá sa kotúľa vnútri inej kružnice bez kĺzania,
- **cykloidy** – krivky tvorené trajektóriou pevného bodu kruhu, ktorý sa kotúľa po priamke bez kĺzania,
- **cyklogóny** – krivky tvorené trajektóriou pevného bodu mnohouholníka, ktorý sa prevaľuje po priamke bez kĺzania.

Obr. 11 Ukážky hypocykloidy, hypotrochoidy, cykloidy a cyklogóny

Námety ďalších úloh na simuláciu fyzikálnych pohybov

Hra zasiahni terč

Vo zvislom smere sa po úsečke pohybuje terč rovnomerným priamočiarym pohybom, pričom na okrajoch úsečky terč zmení smer pohybu na opačný. Vo vodorovnom smere je akustické delo, pomocou ktorého môžeme zasiahnuť pohybujúci sa terč. Navrhnite a naprogramujte hru, v ktorej budete zasahovať kolmo sa pohybujúci terč.

Simulácia pohybu biliardovej gule na obdĺžnikovom stole

Vytvorte program vykresľujúci trajektóriu biliardovej gule pohybujúcej sa rovnomerne priamočiaro na uzavretom obdĺžnikovom stole. Predpokladáme, že biliardová guľa sa pohybuje rovnomerným priamočiarym pohybom bez rotácie, pričom jej odrazy od okrajov stola sú pružné (platí, že uhol dopadu na okraj stola je rovný uhlu odrazu od okraja stola). Ako by sa zmenilo riešenie úlohy, ak by stôl nebol obdĺžnikový, ale kruhový alebo eliptický?

Hod lopty do basketbalového koša

Vytvorte program, ktorý pre náhodne vygenerovanú vzdialenosť hráča od koša a zadanú výšku hráča umožní hráčovi čo najmenším počtom hodov trafiť loptu do koša. Hráč pri každom hode zadáva rýchlosť lopty a uhol, pod ktorým hádže loptu do koša.

Krivka sledovania

Korisť sa pohybuje rovnomerným priamočiarym pohybom. Lovec ju prenasleduje rovnomernou rýchlosťou, pričom je na ňu stále nasmerovaný. Spresnite špecifikáciu tohto problému a naprogramujte trajektóriu pohybu koristi a lovca, ktorý ju prenasleduje.

Pristátie na Mesiaci

Pilot padá s pristávacím modulom z určitej výšky na povrch Mesiaca. Voľný pád modulu môže pilot brzdiť zapnutím motorom proti smeru voľného pádu. Pilot má k dispozícii len obmedzené množstvo paliva. Podľa rýchlosti, ktorú dosiahne modul na povrchu Mesiaca, môžeme konštatovať mäkké pristátie, tvrdé pristátie, resp. deštrukciu modulu.

Jednoduchý letecký simulátor

Naprogramujte jednoduchý letecký simulátor, ktorý bude zohľadňovať gravitačnú silu, silu odporu prostredia, silu vetra zo zadaného smeru, vztlakovú silu a tiež ťah motora lietadla.

Čo sme sa naučili

- Zostaviť analytický, resp. diferenčný model pohybujúceho sa bodu podľa fyzikálnych zákonov,
- naprogramovať simuláciu pohybujúceho sa bodu využívajúcu grafické rozhranie s knižnicou Tkinter s ovládacími prvkami,
- použiť simulácie pohybujúceho sa bodu na overenie, resp. vyvrátenie hypotéz.

Bibliografia

Khan Academy. (2018). *Discovery of the electron and nucleus*. Retrieved from: https://www.khanacademy.org/science/chemistry/electronic-structure-of-atoms/history-of-atomic-structure/a/discovery-of-the-electron-and-nucleus

Projectile motion. (2018). Retrieved from: https://en.wikipedia.org/wiki/Projectile_motion#Trajectory_of_a_projectile_with_air_resistance

Rouinfar, A. (2018). *Projectile Motion - Kinematics | Air Resistance | Parabolic Curve - PhET Interactive Simulations*. Retrieved from: https://phet.colorado.edu/en/simulation/projectile-motion

Šnábl, I. (2018). *Matematická biologie učebnice: Psí křivka*. Retrieved from: http://portal.matematickabiologie.cz/index.php?pg=analyza-a-modelovani-dynamickych-biologickych-dat--spojite-deterministicke-modely-i--nektere-klasicke-ulohy--psi-krivka

Wikipedia. (2017). *Cyclogone*. Retrieved from: https://en.wikipedia.org/wiki/Cyclogon

Wikipedia. (2018). *Cycloid*. Retrieved from: https://en.wikipedia.org/wiki/Cycloid

Wikipedia. (2018). *Epicycloid*. Retrieved from: https://en.wikipedia.org/wiki/Epicycloid

Wikipedia. (2018). *Hypocycloid*. Retrieved from: https://en.wikipedia.org/wiki/Hypocycloid

Wikipedia. (2018). *Hypotrochoid*. Retrieved from: https://en.wikipedia.org/wiki/Hypotrochoid

Register pojmov

A

analytický model, 1

D

diferenčný model, 1, 12

F

fáza, 9

G

generátor pseudonáhodných čísel, 1, 11

K

krivka
- balistická, 1
- cyklogona, 1, 11
- cykloida, 1, 11
- epicykloida, 1, 10, 11
- hypocykloida, 1, 11
- hypotrochoida, 1, 11
- Lissajousova, 1, 9, 10
- sledovania, 11

O

ovládací prvok
- Buttoon, 1, 3
- Canvas, 1, 3
- Entry, 1, 3
- Label, 1, 3
- OptionMenu, 1, 7

P

pohyb
- rovnomerne zrýchlený, 5, 6, 8, 10
- rovnomerný priamočiary, 2, 3, 4

R

rýchlosť, 1, 3, 5, 6, 8, 9, 11, 12
- uhlová, 9, 10

S

sila
- gravitačná, 6, 8, 10
- odporu prostredia, 6, 8, 10

simulácia, 1, 7, 8, 9, 11, 12

T

trajektória, 1, 10

Z

zákon
- 1. Newtonov, 1
- 2. Newtonov, 1, 6, 7
- Coulombov, 1, 7

zrýchlenie, 1, 5, 6, 7, 8