Numeryczne Wyznaczanie Krzywej Pogoni

RRZ Lab Projekt 2

Franciszek Pietrusiak

31 maja 2025

1 Definicja, Historia i Zastosowania

Krzywa pogoni (ang. Pursuit Curve) to tor ruchu punktu (drapieżnika), który zmierza zawsze w kierunku drugiego punktu (ofiara) poruszającego się po ustalonej krzywej. W roku 1732 Pierre Bouguer jako pierwszy badał ten obiekt w celu określenia krzywej po której statek mógł ścigać inny statek na morzu. Krzywą pogoni zajmował się także Leonardo da Vinci. Krzywą Pogoni można stosować tam, gdzie jeden obiekt chce dogonić drugi obiekt, a zatem znajduje swoje zastosowanie w lotnictwie, systemach naprowadzania rakiet i transporcie.

2 Założenia

W tym projekcie, zakładam, że punkty znajdują się na dwuwymiarowym układzie kartezjańskim. Ofiara startuje z punktu $(d, 0)$ i porusza się ze stałą prędkością, po prostej $x = d$ w kierunku rosnących y. Drapieżnik zaczyna w punkcie (x_0, y_0), $x_0 < d$ i przez cały czas porusza się ze stałą prędkością w kierunku ofiary. Stosunek prędkości drapieżnika do prędkości ofiary wynosi r.

3 Wyprowadzenie równania różniczkowego

W każdym momencie drapieżnik jest w punkcie (x, y), a ofiara w punkcie (d, Y). Skoro ścigający porusza się w kierunku ofiary, to styczna do krzywej pogoni w punkcie (x, y) przechodzi przez punkt (d, Y). A zatem:

$$\frac{dy}{dx} = \frac{Y - y}{d - x}$$

z czego wynika, że:

$$Y = (d - x) \cdot \frac{dy}{dx} + y$$

Długość drogi drapieżnika poruszającego się po krzywej wynosi $ds = \sqrt{(dx)^2 + (dy)^2}$, a ofiary dY wtedy pamiętając o stosunku prędkości drapieżnika do prędkości ofiary dostaje:

$$\frac{ds}{dt} = r \cdot \frac{dY}{dt} \implies ds = r \cdot dY \implies \sqrt{(dx)^2 + (dy)^2} = r \cdot dY \implies \sqrt{1 + (\frac{dy}{dx})^2} = r \cdot \frac{dY}{dx}$$

Teraz różniczkując wcześniej obliczone Y po x dostaje:

$$\frac{dY}{dx} = (d - x) \cdot \frac{d^2y}{dx^2}$$

A zatem otrzymuję następujące równanie różniczkowe drugiego stopnia:

$$r \cdot (d - x) \cdot \frac{d^2y}{dx^2} = \sqrt{1 + (\frac{dy}{dx})^2} \quad \text{lub równoważnie} \quad r \cdot (d - x) \cdot y''(x) = \sqrt{1 + (y'(x))^2}$$

4 Układ równań stopnia pierwszego

Chcąc przybliżyć numerycznie rozwiązanie wyprowadzonego wyżej równania drugiego rzędu zamienię je na układ dwóch równań różniczkowych pierwszego rzędu. Niech $y_1(x) = y(x)$ oraz $y_2(x) = y'(x)$. Po prostych przekształceniach dostaje układ:

$$\begin{cases} y_1'(x) = y_2(x) \\ y_2'(x) = \frac{\sqrt{1+(y_2(x))^2}}{r(d-x)} \end{cases}$$

Z warunkami początkowymi:

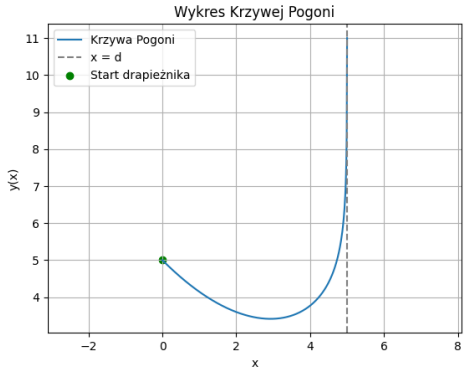
$$y_1(x_0) = y_0 \quad \text{oraz} \quad y_2(x_0) = \frac{-y_0}{d - x_0}$$

5 Numeryczne Rozwiązanie

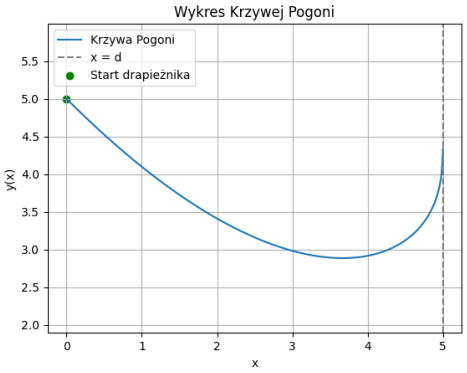
W swoim programie użyłem Metody Rungego-Kutty RK4 do znalezienia przybliżenia krzywej pogoni. Kod programu w Pythonie:

```python
import numpy as np
import matplotlib.pyplot as plt
def krzywa_pogoni(x0, y0, d, r, h=1e-2, max_steps=10000, eps=1e-10, y_limit=1e3):
    x_vals = [x0]
    y_vals = [y0]
    dy_vals = [-y0 / (d - x0)]
    for _ in range(max_steps):
        x = x_vals[-1]
        y1 = y_vals[-1]  # y1 = y(x)
        y2 = dy_vals[-1]  # y2 = y'(x)
        if abs(d - x) < eps: # nie chce dzielic przez zero
            break
        def f1(x, y1, y2): # z pierwszego rownania w powyzszym ukladzie
            return y2
        def f2(x, y1, y2): # z drugiego rownania
            return np.sqrt(1 + y2 ** 2) / (r * (d - x))

        # wyznaczam K1, K2, K3, K4 dla y1 i y2 jednoczesnie
        k1_y1 = h * f1(x, y1, y2)
        k1_y2 = h * f2(x, y1, y2)
        k2_y1 = h * f1(x + h/2, y1 + k1_y1/2, y2 + k1_y2/2)
        k2_y2 = h * f2(x + h/2, y1 + k1_y1/2, y2 + k1_y2/2)
        k3_y1 = h * f1(x + h/2, y1 + k2_y1/2, y2 + k2_y2/2)
        k3_y2 = h * f2(x + h/2, y1 + k2_y1/2, y2 + k2_y2/2)
        k4_y1 = h * f1(x + h, y1 + k3_y1, y2 + k3_y2)
        k4_y2 = h * f2(x + h, y1 + k3_y1, y2 + k3_y2)

        y1 += k1_y1/6 + k2_y1/3 + k3_y1/3 + k4_y1/6
        y2 += k1_y2/6 + k2_y2/3 + k3_y2/3 + k4_y2/6
        x += h
        x_vals.append(x)
        y_vals.append(y1)
        dy_vals.append(y2)
        if abs(y1) > y_limit: # nie chce miec bardzo duzych y w krzywej
            break
    return np.array(x_vals), np.array(y_vals)

x0, y0 = 0, 5
```

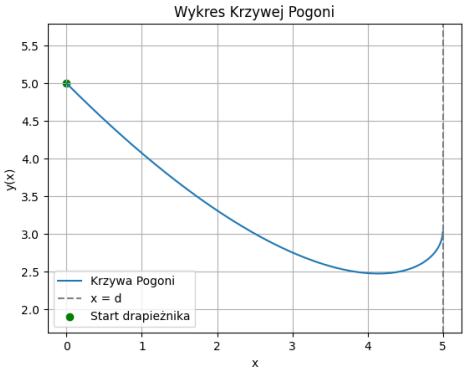
```
d = 5
r = 5
x_vals, y_vals = krzywa_pogoni(x0, y0, d, r)
plt.plot(x_vals, y_vals, label="Krzywa Pogoni")
plt.axvline(x=d, color='gray', linestyle='--', label="x = d")
plt.scatter([x0], [y0], color='green', label='Start drapieznika')
plt.xlabel("x")
plt.ylabel("y(x)")
plt.title("Wykres Krzywej Pogoni")
plt.legend()
plt.grid()
plt.axis('equal')
plt.show()
```



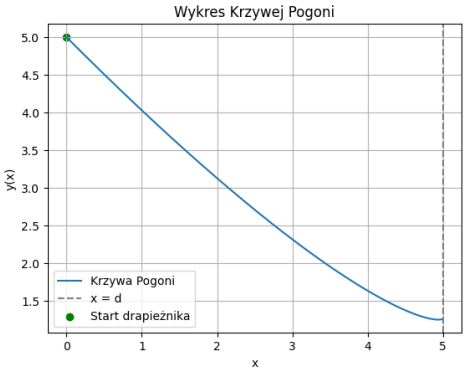
Rysunek 1: $(x_0, y_0) = (0, 5)$, $r = 1$, $d = 5$



Rysunek 2: $(x_0, y_0) = (0, 5)$, $r = 1.5$, $d = 5$



Rysunek 3: $(x_0, y_0) = (0, 5)$, $r = 2$, $d = 5$



Rysunek 4: $(x_0, y_0) = (0, 5)$, $r = 5$, $d = 5$