Univerzita Karlova
Přírodovědecká fakulta

Studijní program: Geografie a kartografie

Nelly Polanská
2. ročník

Úvod do programování

1. Iterativní výpočet odmocniny z čísla c s chybou menší než ε. (27 - matematika)

1.1. Rozbor problému

Iterativní výpočet odmocniny je založený na Newtonově metodě. Ta spočívá v aproximaci funkce $f(x) = 0$ postupností $(x_n)_{n=1}^{\infty}$, kterou lze zadat rekurentně: [1]

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{1}$$

Pokud chceme určit druhou odmocninu čísla c, řešíme rovnici $x^2 = c$. V oboru reálných čísel platí $c > 0$. Abychom mohli použít Newtonovu metodu, zvolíme $f(x) = x^2 - c$ rovnou nule. Dosazením do vztahu 1 dostáváme:

$$x_{n+1} = x_n - \frac{{x_n}^2 - c}{2x_n} \tag{2}$$

Funkci $f(x)$ budeme aproximovat posloupností s počátečním členem $x_0 = c$, vztahem číslo 2, až dokud rozdíl mezi dvěma libovolnými sousedními členy postupnosti nebude menší než zvolená chyba ε.

1.2. Struktura programu a zvolené algoritmy

Odmocninu z čísla c můžeme vypočítat například použitím knihovny NumPy, funkcí $sqrt$, a nebo jednoduše dosadit $c^{0,5}$. Tyto triviální metody v praxi fungují, avšak na iterativní výpočet odmocniny i s chybou použijeme objektově orientované programování, konkrétně třídu $SqrtCalculator$ a konstruktor $\_\_init\_\_$, kterým zadáme parametry c a ε. Aby mohla být použita Newtonova metoda na výpočet odmocniny čísla c v oboru reálných čísel, musíme programu zadat podmínku $c > 0$, v opačném případě funkce nemá řešení, což vyjádříme pomocí $raise\ ValueError$. Stejně tak by číslo mělo být větší než chyba, proto pokud zadáme chybu větší než samotné číslo, program vypíše chybu. Bylo by možné, místo $raise\ ValueError$ použít jednodušší variantu, a to vypsat chybu pomocí $print$. Pokud je podmínka splněna, ukládáme hodnoty c a ε jako atributy objektu pomocí $self$ (kterou musíme také zadefinovat v konstruktoru).

Následně zadefinujeme samotný výpočet. Vytvoříme cyklus, který bude aproximovat funkci posloupností s počátečním členem $x_0 = c$, vztahem číslo 2, až dokud rozdíl mezi dvěma libovolnými sousedními členy posloupnosti nebude menší než zvolená chyba ε. Aby cyklus pokračoval v aproximaci, musí platit podmínka:

$$|x^2 - c\ | > \varepsilon \tag{3}$$

Podmínku programu zadáme pomocí $while$. Cyklus začne na hodnotě $x = c$ a bude postupně pomocí vztahu 2 hledat také x, pro které bude splněna podmínka 3. Když tuto hodnotu najde, pomocí $return$ nám vrátí vypočítanou hodnotu odmocniny. Pokud bychom chtěli odmocninu vypočítat přesně, volíme $\varepsilon = 0$.

2. Výpočet souřadnic průsečíku dvojice úseček (60 – počítačová geometrie)

Na reprezentaci bodů v dvojrozměrném prostoru použijeme objektově orientované programování, konkrétně třídu $Point$, která nám pomůže při uchovávání souřadnic a jejich reprezentaci. V ní zadefinujeme konstruktor $\_\_init\_\_$, který bude pracovat se souřadnicemi bodů úseček. Abychom souřadnice mohli vypisovat ve formátu (x, y), použijeme $\_\_repr\_\_$. Tento krok není úplně nezbytný, ale zlepší nám přehlednost programu. Jako proměnné na uložení souřadnic bodu (x, y) používáme $self.x$ a $self.y$. Tuto metodu nám pomohla navrhnout umělá inteligence. Taktéž v programu použijeme anotaci $float$, aby uměl pracovat i s desetinnými hodnotami souřadnic.

Také zadefinujeme metodu $\_\_repr\_\_$, pomocí které program vrací výstup objektu třídy Point. Pro nejefektivnější formátování řetězců použijeme f-string. Alternativně by bylo možné použít i spojování řetězců nebo $str.format$. Spojování řetězců by po přeformátování souřadnic na str fungovalo, avšak je to velmi nepraktické, protože by obsahoval velké množství sčítacích operací, což by se při dvou úsečkách a čtyřech bodech ještě dalo, ale při větším počtu bodů by se tato metoda ukázala jako velmi repetetivní a neúčinná, F-string jsme zvolili kvůli efektivitě i přehlednosti – ukládá proměnné přímo v řetězci, což zlepšuje čitelnost a přehlednost programu.

Na reprezentaci samotné úsečky pomocí dvou bodů třídy Point použijeme třídu LineSegment. Abychom mohli počítat vzájemné polohy úseček, potřebujeme jejich směrový vektor, který vypočítáme jako rozdíl souřadnic koncového a počátečního bodu. Směrový vektor označíme v. Budeme ho následně používat na určení vzájemné polohy.

Následně vytvoříme funkci, která na základě porovnání směrových vektorů určí, zda jsou dané úsečky rovnoběžné, což platí, pokud jsou jejich směrové vektory násobky toho stejného vektoru. Funkci nazveme $rovnobeznost$. Úsečky jsou rovnoběžné, pokud platí nulová hodnota determinantu:

$$\det = dx_1 \cdot dy_2 - dx_2 \cdot dy_1 = 0$$

Kde (dx_1, dy_1) a (dx_2, dy_2) jsou parametry daného směrového vektoru. Pokud je tato podmínka splněna, může taktéž nastat, že jsou úsečky souběžné. Zadefinujeme proto funkci bod, která zjistí, zda rovnoběžné úsečky sdílejí alespoň jeden společný bod. Funkci bod nám pomohla navrhnout umělá inteligence. Pokud dvě rovnoběžné úsečky společný bod sdílejí, jsou souběžné, pokud ne, jsou rovnoběžné.

Pokud dvě úsečky rovnoběžné nebudou, můžou se protínat, za předpokladu, že hodnota determinantu je nenulová. Na výpočet průsečíku použijeme zadefinovanou funkci $prusecik$. Průsečík úseček určíme řešením soustavy dvou lineárních rovnic s parametry t a u:

$$A_x + t\,dx_1 = C_x + u\,dx_2$$

$$A_y + t\,dy_1 = C_y + u\,dy_2$$

Kde, díky nenulovosti determinantu použitím Cramerova pravidla [4] získame hodnoty t a u vyčíslené jako:

$$t = \frac{(C_x - A_x)\,dy_2 - (C_y - A_y)\,dx_2}{dx_1 \cdot dy_2 - dx_2 \cdot dy_1}$$

$$u = \frac{(C_x - A_x)\,dy_1 - (C_y - A_y)\,dx_1}{dx_1 \cdot dy_2 - dx_2 \cdot dy_1}$$

Kde $u, t \in \langle 0,1 \rangle$ (v opačném případe jsou úsečky různoběžné, ale neprotínají se). Vyčíslené hodnoty následně dosadíme do původních rovnic úsečky. Hodnoty obou parametrů musíme vypočítat, abychom ověřili podmínku $u, t \in \langle 0,1 \rangle$, avšak při výpočtu lineární rovnice nám stačí použít jen jeden z parametrů, v našem případě konkrétně t.

Po zhotovení vypsaných modulů se zaměříme na hlavní program, označený $main$. Na začátku mu zadáme body A, B, C a D, ve tvaru (x, y), který jsme deklarovali na začátku třídou $Point$. Pomocí LineSegment a bodů A, B, C, D si zadefinujeme úsečky p a q. Následně, prostřednictvím základních logických operací if, elif a else pracujeme s usečkami. Začneme tím, že pomocí funkce soubeznost zjistíme, zda jsou úsečky souběžné. Pokud ano, program vypíše, že jsou, a skončí. Pokud ne, funkcí rovnoběžnost zjistíme, zda jsou rovnoběžné. Pokud rovnoběžné nejsou, aplikuje se funkce průsečík, a program zjistí, zda mají společný bod. Pokud ano, program nám vypočítá i souřadnice průsečíku ve tvaru (x, y). V opačném případě napíše, že jsou různoběžné bez průsečíku.

Zdroje:

https://kam.fit.cvut.cz/deploy/bi-zma/mirror/textbook/sec_newtonova_metoda.html [1]

https://stackoverflow.com/questions/70793490/how-do-i-calculate-square-root-in-python [2]

https://www.svf.stuba.sk/buxus/docs/dokumenty/skripta/Algebra_Skripta_V.pdf [3]

https://cs.wikipedia.org/wiki/Cramerovo_pravidlo [4]