Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Interaktívne vyšetrovanie priebehu elementárnych funkcií

Bakalárska práca

2020

Juraj Vetrák

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

INTERAKTÍVNE VYŠETROVANIE PRIEBEHU ELEMENTÁRNYCH FUNKCIÍ

BAKALÁRSKA PRÁCA

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Ing. Ján Komara, PhD.

Bratislava, 2020
Juraj Vetrák

23814387

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Juraj Vetrák
Študijný program:	aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	informatika
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Interaktívne vyšetrovanie priebehu elementárnych funkcií
Properties of Elementary Functions Interactively

Anotácia: Návrh, vývoj a implementácia editora pre interaktívne vyšetrovanie priebehu elementárnych funkcií vo výpočtovom prostredí IPython/Jupyter.

Vedúci:	Ing. Ján Komara, PhD.
Katedra:	FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry:	prof. Ing. Igor Farkaš, Dr.
Dátum zadania:	20.09.2019

Dátum schválenia: 07.10.2019

doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.... študent

.... vedúci práce

Poďakovanie: ...

Abstrakt

...

Kľúčové slová: ...

Abstract

...

Keywords:

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

TODO

1 Motivácia

S konceptom vyšetrovania priebehu funkcií pomocou diferenciálneho počtu sa študenti vysokých škôl stretnú väčšinou až v neskorších fázach úvodného kurzu matematickej analýzy. S rôznymi funkciami, aj zložitejšími, sa však študenti stretnú oveľa skôr, pričom sú schopní nájsť niektoré kvantitatívne charakteristiky týchto funkcií len pomocou skúmania ich grafu. V mnohých stredoškolských študijných materiáloch sú totiž tieto grafy uvedené spolu s predpismi ich funkcií a ak nie, už základná programátorská zručnosť umožňuje študentom vykreslenie týchto grafov na obrazovkách svojich počítačov.

Pod vyšetrovaním priebehu funkcie sa rozumie nájdenie takých vlastností funkcie, ktoré sú potrebné pre čo najpresnejšie nakreslenie jej grafu. Týmito vlastnosťami sú podľa autorov skrípt [8, str. 113], či učebnice [7, str. 395] najmä:

1. definičný obor funkcie;
2. spojitosť funkcie a jej správanie sa v bodoch nespojitosti
3. nulové body funkcie;
4. body, v ktorých funkcia nadobúda extrémy a intervaly, na ktorých je funkcia monotónna;
5. inflexné body a intervaly, na ktorých je funkcia konvexná alebo konkávna;
6. asymptoty grafu funkcie.

Hlavnou motiváciou pre vznik tejto práce je potreba vytvorenia nástroja, ktorý slúži ako pomocník študentov pri vyšetrovaní priebehu funkcie, a to najmä na začiatku ich štúdia na vysokej škole, keď sa ešte nepredpokladá ich znalosť diferenciálneho počtu. Nástroj má za úlohu umožniť rýchle, interaktívne a presnejšie vyšetrenie tých vlastností priebehu funkcie, ktoré je možné graficky vizualizovať - definičný obor, nulové body, extrémy, intervaly monotónnosti, inflexné body, či intervaly konvexnosti a konkávnosti.

Obsah tejto kapitoly je rozdelený na tri sekcie popisujúce využitie rôznych prostriedkov na vyšetrovanie priebehu funkcie. Postupnosť obsahu týchto častí je uvažovaná ako priamo úmerná s pribúdajúcimi znalosťami študenta v úvodnom kurze matematickej analýzy.

Vo všetkých troch sekciách sa pre čo najväčšiu mieru uniformnosti popisuje vyšetrovanie niektorých vlastností elementárnej funkcie

$$f(x) = \frac{x^2}{2} - 2x - \frac{\ln(x^2+1)}{2} + 3\arctan x, \tag{1}$$

na vybranej časti jej definičného oboru.

1.1 Vyšetrovanie pomocou tabuľky funkčných hodnôt

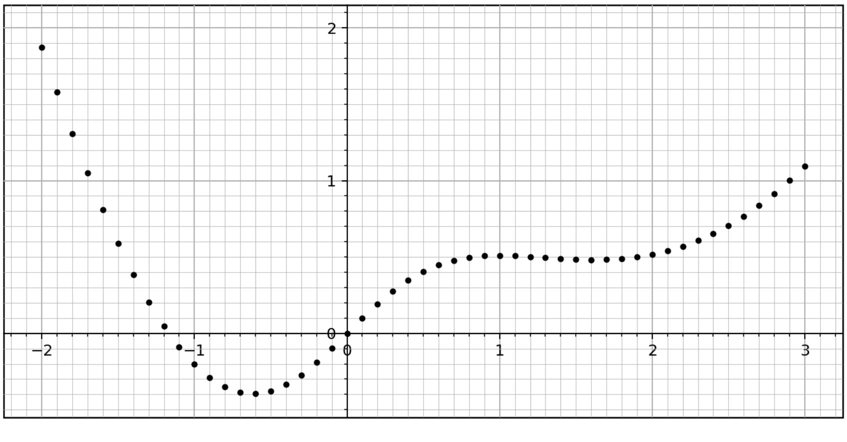
Nech f je funkcia definovaná vzťahom (1). Intuitívnym spôsobom, ako nájsť niektoré vlastnosti priebehu tejto funkcie, je nakreslenie jej grafu. Pod nakreslením grafu funkcie sa uvažuje vyznačenie bodov na súradnicovú os, ktoré vznikli nanesením vybraných hodnôt nezávislej premennej x na jej horizontálnu časť a funkčných hodnôt (hodnôt závislej premennej y) na jej vertikálnu časť. **UNFINISHED** DAVA TO ZMYSEL?

Funkciu f je možné vyjadriť prehľadným spôsobom vo forme tabuľky hodnôt nezávislej premennej x a príslušných funkčných hodnôt (zaokrúhlených na dve desatinné miesta) z vybranej časti definičného oboru funkcie f.

x	-2.0	-1.9	-1.8	...	2.8	2.9	3.0
$f(x)$	1.87	1.58	1.30	...	0.91	1.00	1.10

Poznámka. Desatinné čísla sú z dôvodu väčšej prehľadnosti textu uvedené v notácii s desatinnou bodkou, namiesto desatinnej čiarky.

Jednoduchý graf funkcie f je tak možné zostrojiť nanesením jednotlivých stĺpcov tabuľky v podobe bodov roviny na milimetrový papier (obr. 1.1).



Obr. 1.1: Graf funkcie - milimetrový papier

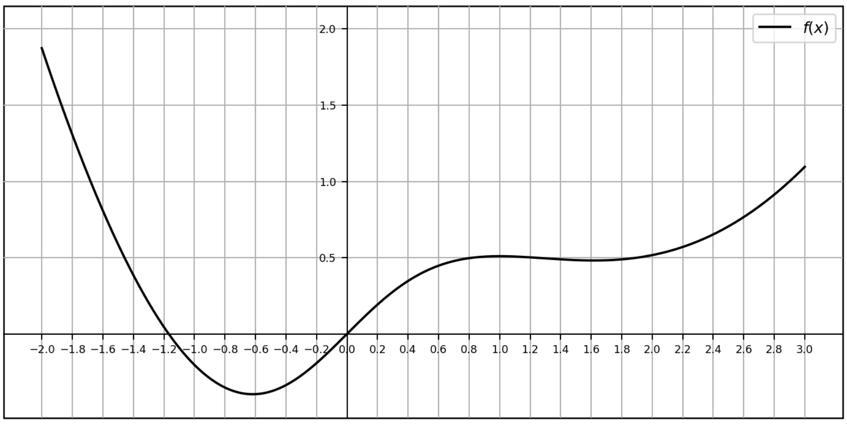
Určenie niektorých kvantitatívnych charakteristík z grafu funkcie f na obrázku 1.1 nie je vôbec zložité. Z vybraných vlastností priebehu funkcie, uvedených na začiatku tejto kapitoly, vie študent napríklad určiť jeden nulový bod v bode $x = 0$. Rovnako vie len z detailného pohľadu na tento graf určiť, že funkcia nadobúda extrémy v bodoch $\{-0.6, 1, 1.6\}$ a pozorovať tak aj intervaly monotónnosti, ktoré tieto body ohraničujú.

Nedá sa však nepozorovať, že funkcia f má aj druhý nulový bod, pričom je možné vidieť, že sa nachádza niekde v tretej tretine intervalu $(-2, -1)$. Táto informácia je však v snahe určiť čo najpresnejšiu hodnotu tohto bodu nepostačujúca. Využitie milimetrového papiera na zostrojenie grafu funkcie má tak značné obmedzenia pri snahe nájsť presnejšie hodnoty na súradnicových osiach. Študent si však už pri základnej vedomosti programovacieho jazyka Python môže dopomôcť vykreslením grafu funkcie pomocou knižnice matplotlib a vylepšiť si tak pozorovaciu schopnosť napr. vhodným prispôsobením delenia osi x.

```
In [1]: import matplotlib.pyplot as plt, numpy as np
        def f(X): return X**2/2-2*X-np.log(np.sqrt(X**2+1))+3*np.arctan(X)
        X = np.linspace(-2, 3, 5*100+1)
        fig, ax = plt.subplots(figsize=(10, 4))
        d = 0.2
        ax.set_xticks(np.arange(-2, 3+d, d))
        ax.plot(X, f(X), c='black', label='$f(x)$')
        ax.grid(); ax.legend();
        fig.show()
```

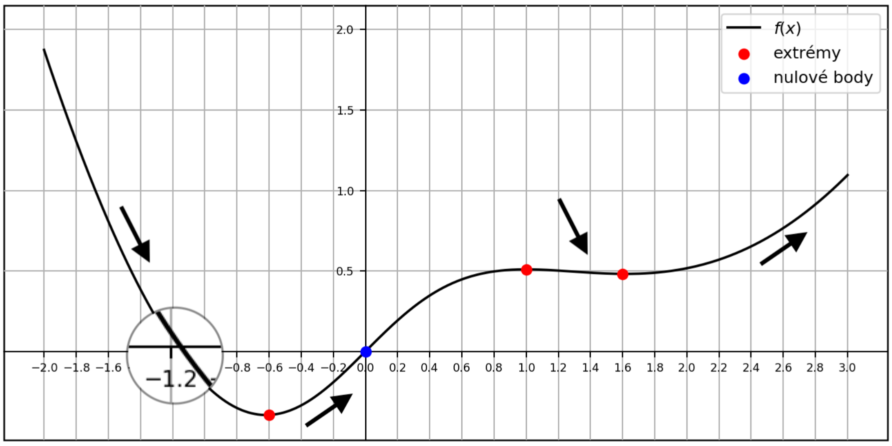
Uvedený je príklad kódu, kde `f` je definícia funkcie f určenej vzťahom (1), `X` je uniformné delenie základného intervalu a príkaz `ax.set_xticks(np.arange(-2, 3+d, d))` znamená zjemnenie zobrazovaného delenia osi x na dieliky vzdialené `d` $= \Delta x = 0.2$.

Po spustení sa vykreslí graf viditeľný na obr. 1.2, pričom je prostredníctvom neho možné overiť správnosť nameraných hodnôt z milimetrového papiera (1.1).



Obr. 1.2: Graf funkcie - grafický softvér

Určenie hľadaného nulového bodu sa spresnilo len na interval $(-1.2, -1)$, čo stále nie je dostatočné presná informácia. Priblíženie sa k skutočnej hodnote tohto bodu môže znamenať ďalšie zmenšovanie hodnoty Δx. Tento postup však prestáva byť po čase prehľadný a efektívny, pričom aj v prípade už odhadnutých význačných bodov, či intervalov, nezaručuje ich skutočnú správnosť. Vyriešiť tento problém má za cieľ táto bakalárska práca, konkrétne jej programová časť - interaktívny editor. Jeho použitie popisuje nasledujúca sekcia tejto kapitoly.



Obr. 1.3: Graf funkcie - možné riešenie

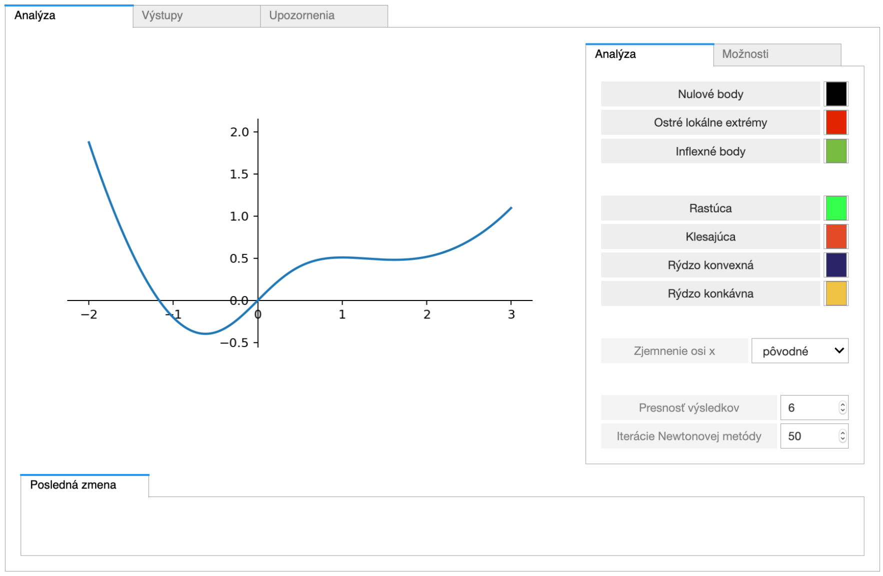
1.2 Vyšetrovanie pomocou interaktívneho editora

Vyšetrovanie priebehu funkcie len pomocou tabuľky funkčných hodnôt, resp. grafu funkcie v sekcii 1.1 sa ukázalo ako vcelku intuitívne, no nekompletné a s výrazným rizikom nepresností. Ďalším prostriedkom na vyšetrovanie priebehu funkcie, ktorý si ponecháva prvky intuitívnosti, pričom do značnej miery odstraňuje problémy súvisiace s nekompletnosťou a nepresnosťou výsledkov, je interaktívny editor - programová časť tejto práce.

Obsahom tejto sekcie je názorná ukážka jeho funkcionality, ktorá umožňuje vypočítať a vizualizovať výsledky takpovediac ihneď, a to bez potreby rozsiahlejších programátorských zručností.

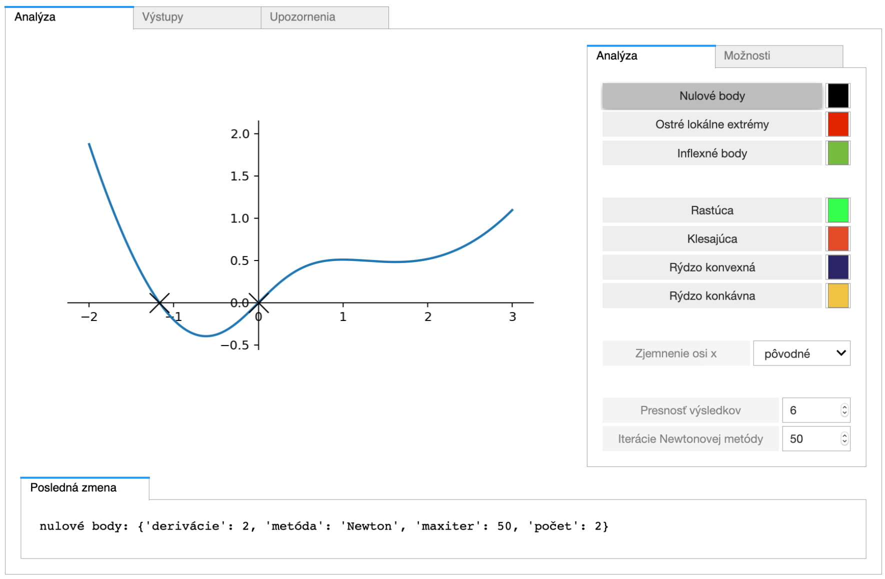
Spustenie editora prebieha pridaním nasledovného riadku do kódu na vykresľovanie grafu zo sekcie 1.1.

```
In [1]: editor(function=f, figure=fig, axes=ax, intervals=[X])
```



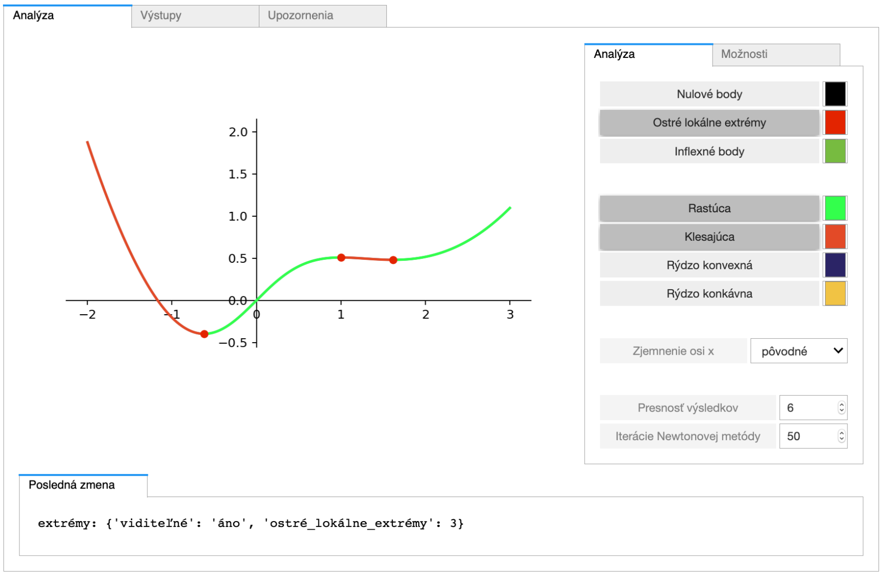
Obr. 1.4: Editor - úvodná obrazovka

Obrázky 1.5 a 1.6 demonštrujú jednoduché použitie editora vybratím príslušných vlastností priebehu funkcie v hlavnom menu. Editor po zvolení možnosti nulových bodov tieto body ihneď vykreslí (obr 1.5), pričom je značne presnejší ako ľudské oko. Program totiž vo svojom výstupe ponúka vypočítané hodnoty týchto bodov (v notácii hranatých zátvoriek reprezentujúcich dátovú štruktúru `list` jazyka Python) s možnosťou zvýšenej presnosti výsledkov:



Obr. 1.5: Editor - nulové body

```
Out [1]: nulove_body: [-1.16822, 0.0]
Out [2]: nulove_body: [-1.16822492541, 0.0]
```



Obr. 1.6: Editor - monotónnosť a extrémy

Veľmi podobné je to pri hľadaní ostrých lokálnych extrémov, či intervalov monotónnosti. V tomto prípade ich editor tiež ihneď vizualizuje (obr. 1.6) a ponúkne svoje riešenia. V prípade ostrých lokálnych extrémov sú to body:

```
Out [1]: ostre_lokalne_extremy_v_bodoch: [-0.62, 1.0, 1.62]
Out [2]: ostre_lokalne_minima_v_bodoch: [-0.62, 1.62]
Out [3]: ostre_lokalne_maxima_v_bodoch: [1.0]
```

Editor zároveň našiel aj intervaly monotónnosti:

```
Out [1]: klesajuca_intervaly_x: [(-2.0, -0.63), (1.01, 1.61)]
Out [2]: rastuca_intervaly_x: [(-0.61, 0.99), (1.63, 3.0)]
```

1.3 Vyšetrovanie pomocou derivácií

Na vyšetrovanie priebehu funkcie sa v neskorších etapách vysokoškolského štúdia matematickej analýzy používajú silnejšie prostriedky, ako napríklad diferenciálny počet. Vyšetrovanie priebehu funkcie pomocou diferenciálneho počtu je obvyklý matematický postup odvolávajúci sa na definície a vety zo sekcie 2.2 tejto práce, resp. ďalšie pojmy týkajúce sa diferenciálneho počtu z citovanej literatúry od autorov Kubáček a Valášek [8].

Vyšetrovanie niektorých vlastností pomocou týmto spôsobom je taktiež demonštrované na príklade funkcie f, definovanej vzťahom (1). Ako je spomenuté na záver, aj pri tomto postupe nájde výstupný program tejto práce využitie. Vzhľadom na čo najväčšiu elimináciu chýb sú výpočty niektorých hodnôt kontrolované programom WolframAlpha [18], resp. Python knižnicou pre symbolickú matematiku SymPy [16].

Definičný obor. Výraz $\frac{x^2}{2} - 2x - \frac{\ln(x^2+1)}{2} + 3\arctan x$, ktorým je funkcia f definovaná, má zmysel pre všetky reálne čísla x. Definičným oborom je teda množina reálnych čísel $\mathbb{R}$ s krajnými bodmi $-\infty$ a $+\infty$, na ktorej je táto funkcia spojitá.

Extrémy a intervaly monotónnosti.

Veta 1. *Ak funkcia f má lokálny extrém vo vnútornom bode c svojho definičného oboru, tak buď neexistuje $f'(c)$, alebo $f'(c) = 0$. Bod c sa nazýva stacionárny bod funkcie f, ak $f'(c) = 0$. Pri hľadaní lokálnych extrémov funkcie f treba teda vyšetriť:*

1. *všetky jej stacionárne body;*
2. *všetky body definičného oboru funkcie f, v ktorých neexistuje $f'(c)$;*
3. *všetky body definičného oboru funkcie f, ktoré nie sú jeho vnútornými bodmi.*

Podľa uvedenej vety 1 zo skrípt od Kubáčka a Valáška [8], je pre nájdenie lokálnych extrémov funkcie f potrebné uviesť definíciu jej prvej derivácie,

$$f'(x) = \frac{(x-1)(x^2-x-1)}{x^2+1}.$$

Nech V_1 je množina všetkých stacionárnych bodov funkcie f. Vypočítaním rovníc $x-1=0$ a $x^2-x-1=0$ je možné nájsť stacionárne body funkcie f, a teda

$$V_1 = \left\{ \frac{1-\sqrt{5}}{2}, 1, \frac{1+\sqrt{5}}{2} \right\}.$$

Nech V_2 je množina všetkých bodov definičného oboru funkcie f, v ktorých nemá f deriváciu. Keďže pre všetky x z definičného oboru platí $x^2+1>0$,

$$V_2 = \varnothing.$$

Nech V_3 je množina všetkých bodov definičného oboru funkcie f, ktoré nie sú jeho vnútornými bodmi. Potom

$$V_3 = \{-\infty, +\infty\}.$$

Nech $V = V_1 \cup V_2 \cup V_3$. Usporiadanie prvkov množiny V podľa veľkosti

$$-\infty < \frac{1-\sqrt{5}}{2} < 1 < \frac{1+\sqrt{5}}{2} < +\infty$$

dáva istý rozklad definičného oboru funkcie f. Z tohto rozkladu ihneď vyplýva, že prvá derivácia funkcie f existuje a je nenulová v každom bode intervalov $(-\infty, \frac{1-\sqrt{5}}{2})$, $(\frac{1-\sqrt{5}}{2}, 1)$, $(1, \frac{1+\sqrt{5}}{2})$, $(\frac{1+\sqrt{5}}{2}, +\infty)$.

Toto poznanie umožňuje určiť znamienko prvej derivácie funkcie f v jednotlivých bodoch daných intervalov a použiť tak vetu 4 z časti 2.2.2 týkajúcu sa intervalov monotónnosti. Rastúca je teda na intervaloch $(\frac{1-\sqrt{5}}{2}, 1)$, $(\frac{1+\sqrt{5}}{2}, +\infty)$ a klesajúca na intervaloch $(-\infty, \frac{1-\sqrt{5}}{2})$, $(1, \frac{1+\sqrt{5}}{2})$.

Podľa vety 2 z časti 2.2.2 je možné použiť druhú deriváciu funkcie f,

$$f''(x) = \frac{x(x^3+3x-6)}{(x^2+1)^2},$$

na charakterizovanie bodov množiny V_1 ako ostrých lokálnych miním alebo ostrých lokálnych maxím, pretože pre každý bod x z tejto množiny platí $f'(x) = 0$. Keďže $f''(1) < 0$, funkcia f má podľa tejto vety v bode 1 ostré lokálne maximum. Keďže $f''(\frac{1-\sqrt{5}}{2}) > 0$ aj $f''(\frac{1+\sqrt{5}}{2}) > 0$, funkcia f má v bodoch $\frac{1-\sqrt{5}}{2}$ a $\frac{1+\sqrt{5}}{2}$ ostré lokálne minimá.

Inflexné body a intervaly konvexnosti a konkávnosti. Už uvedenú druhú deriváciu funkcie f je podľa vety 6 z časti 2.2.2 možné využiť na určenie intervalov rýdzej konvexnosti a rýdzej konkávnosti.

Nech V_1 je množina všetkých bodov definičného oboru funkcie f, v ktorých je druhá derivácia funkcie f nulová. Potom

$$V_1 = \left\{0, \frac{(3+\sqrt{10})^{\frac{2}{3}} - 1}{\sqrt[3]{3+\sqrt{10}}}\right\},$$

pričom z dôvodu väčšej prehľadnosti textu je v nasledujúcich zápisoch zlomok $\frac{(3+\sqrt{10})^{\frac{2}{3}}-1}{\sqrt[3]{3+\sqrt{10}}}$ vyjadrený jeho približnou hodnotou 1.29.

Nech V_2 je množina všetkých bodov definičného oboru funkcie f, ktoré nie sú jeho vnútornými bodmi. Potom

$$V_2 = \{-\infty, +\infty\}.$$

Rovnakým postupom ako pri intervaloch monotónnosti, teda usporiadaním bodov $V_1 \cup V_2$ podľa veľkosti,

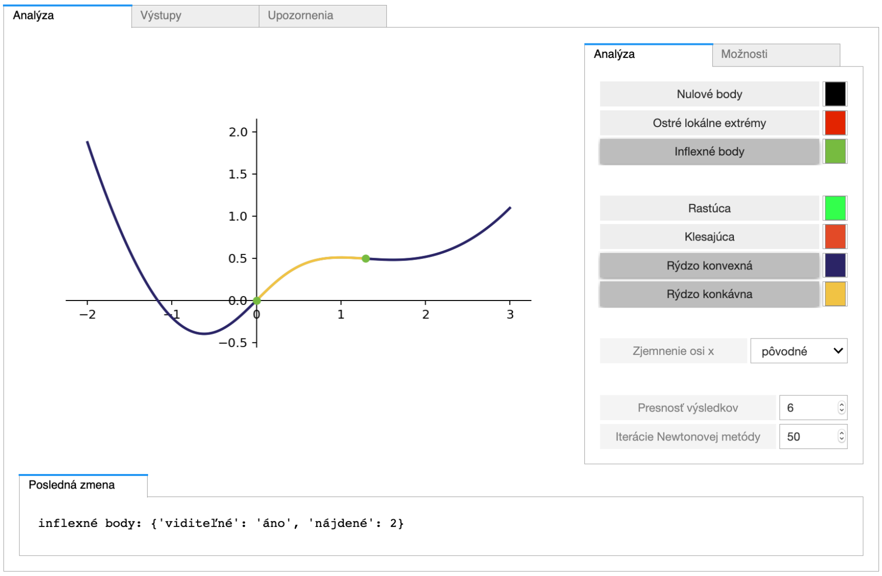
$$-\infty < 0 < 1.29 < +\infty,$$

je možné nájsť intervaly $(-\infty, 0)$, $(0, 1.29)$, $(1.29, \infty)$, v ktorých je funkcia f rýdzo konvexná alebo rýdzo konkávna. Toto poznanie umožňuje určiť znamienko druhej derivácie funkcie f v jednotlivých bodoch daných intervalov a klasifikovať tak podľa vety 6 z časti 2.2.2 intervaly $(-\infty, 0)$, $(1.29, \infty)$ ako intervaly, na ktorých je f rýdzo konvexná a $(0, 1.29)$ ako interval, na ktorom je f rýdzo konkávna.

Aby sa body množiny V_1 mohli nazvať inflexnými, je potrebné podľa vety 5 z časti 2.2.2 overiť, či je tretia derivácia funkcie f v týchto bodoch nenulová. Výpočet tretej derivácie je však oproti prvým dvom pomerne netriviálny, a tak môže byť ponechaný interaktívnemu editoru, ktorý si ju dopočíta numericky a vyhodnotí tak namiesto užívateľa, či tieto body spĺňajú danú podmienku. Ako je vidieť na obrázku 1.7, editor tieto body naozaj vyhodnotil ako inflexné, pričom ponúkol výstup, ktorý sa zhoduje s výpočtami vyššie.

```
Out [1]: inflexne_body: [0.0, 1.29]
```

Užitočnosť editora sa tak môže prejaviť aj pri vyšetrovaní priebehu funkcie takýmto matematickým spôsobom, pričom jeho funkcia je najmä kontrolná.



Obr. 1.7: Editor - konvexnosť, konkávnosť a inflexné body

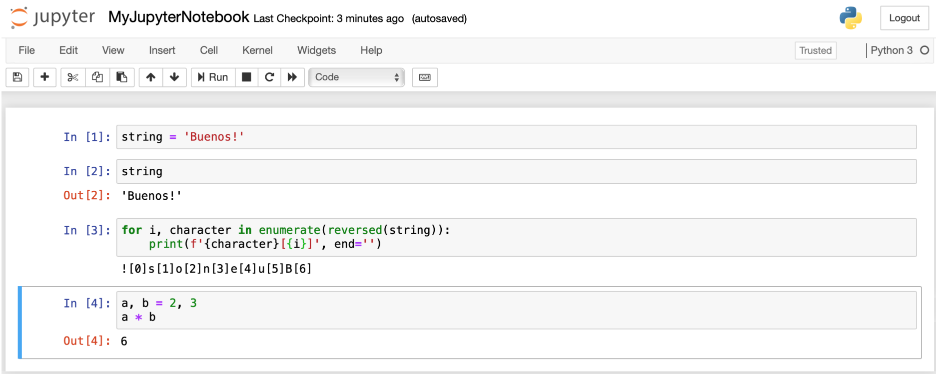
2 Východiská

Pred popisom návrhu a implementácie riešenia, prezentovaného v časti 1.2, je dôležité objasniť východiskový stav pred začatím práce, teda popísať uvažované technológie, programy a knižnice, v ktorých bude práca vyvíjaná (sekcia 2.1), a zároveň popísať základnú, v tomto prípade najmä matematickú teóriu, ktorá objasní funkcionalitu niekoľkých algoritmov použitých pre celkovú funkčnosť riešenia (sekcia 2.2). V sekcii 2.3 je ďalej uvedených niekoľko existujúcich riešení danej problematiky v podobe verejne dostupných open-source programov. V závere tejto kapitoly je popísaná a porovnaná podobná práca, ktorá bola obhájená v roku 2017 (sekcia 2.4).

2.1 Technologické východiská

Editor, všetky jeho funkcionality a výpočtové procesy sú naprogramované v programovacom jazyku Python. Jednou z výhod tohto jazyka je, že ho používa početná komunita vývojárov v rôznych vedeckých oblastiach, ako matematika, dátová veda, či umelá inteligencia. To podporilo vznik mnohých nástrojov a knižníc, ktoré uľahčujú každodennú prácu v spomínaných oblastiach. Niektoré z týchto nástrojov a knižníc tvoria práve jadro tejto bakalárskej práce.

Jupyter Notebook. Okrem programovacieho jazyka Python túto prácu charakterizuje prostredie Jupyter Notebook (obr. 2.1), do ktorého je implementovaná a v ktorom sa používa. Jupyter Notebook autorka praktickej cvičebnice *IPython Cookbook* [14] charakterizuje ako webové interaktívne prostredie, ktoré kombinuje kód, textové prvky, obrázky, videá, animácie, matematické rovnice, grafy, mapy, interaktívne prvky, widgety a grafické užívateľské rozhranie do jediného dokumentu. Spustenie tohto prostredia je realizované príkazom `jupyter notebook` v príkazovom riadku. Práca s týmto nástrojom však nie je limitovaná použitím programovacieho jazyka Python, rôzne numerické výpočty môžu byť totiž realizované aj v iných populárnych jazykoch dátovej vedy, ako Julia, či R. O spúšťanie užívateľského kódu a rôzne numerické výpočty sa starajú programy zvané IPython kernely. IPython je projekt, ktorý vznikol s cieľom poskytnúť nie len značne vylepšený Python shell, ale aj možnosti pre interaktívne, distribuované a paralelné výpočty. Na podobné účely slúžia známe komerčné prostredia ako MATLAB, The Interactive Data Language pre numerické výpočty, Mathematica a Maple pre manipuláciu so symbolmi, či ich otvorené alternatívy GNU Data Language, Octave, Maxima alebo Sage [13].



Obr. 2.1: Interaktívne webové prostredie Jupyter Notebook

NumPy. Prostredie Jupyter Notebook umožňuje využiť výpočtové operácie jazyka Python, no v mnohých prípadoch, ako aj v prípade tejto bakalárskej práce, je zoznam týchto operácii nedostatočný a je potrebné ho rozšíriť o ďalšie numerické operácie. Knižnica NumPy, menej známa ako Numerical Python, je jedným z najdôležitejších balíkov v rámci numerického počítania v jazyku Python.

Medzi príklady funkcionalít knižnice NumPy, ktoré uvádza autor knihy *Python for Data Analysis* [11] a sú kľúčové pre túto prácu, patrí:

- ndarray, efektívne viacrozmerné pole umožňujúce na ňom vykonávať rýchle aritmetické (vektorizované) operácie;
- matematické funkcie pre rýchle operácie nad poliami dát bez potreby písania cyklov;
- podmienený výber dát priamo ako argument polí, namiesto vetiev if-elif-else;
- algoritmy na týchto poliach ako triedenie, výber unikátnych hodnôt, či operácie s množinami.

Autor ďalej vysvetľuje, prečo je knižnica taká dôležitá pre prácu s numerickými výpočtami. Knižnica je totiž priamo navrhnutá na efektívne prácu s veľmi veľkými štruktúrami dát. Aby spomenutú efektívnosť dosiahla, ukladá dáta v súvislých blokoch pamäte, nezávisle od vstavaných objektov jazyka Python. Algoritmy v NumPy sú napísané v jazyku C a vedia pracovať so spomínanou pamäťou bez vyčerpávajúcej kontroly typov jednotlivých údajov. Ďalším, už naznačeným dôvodom je, že NumPy vo väčšine prípadov obchádza štandardné for cykly jazyka Python a namiesto nich vykonáva komplexné výpočty na celých poliach naraz pomocou paralelizmu.

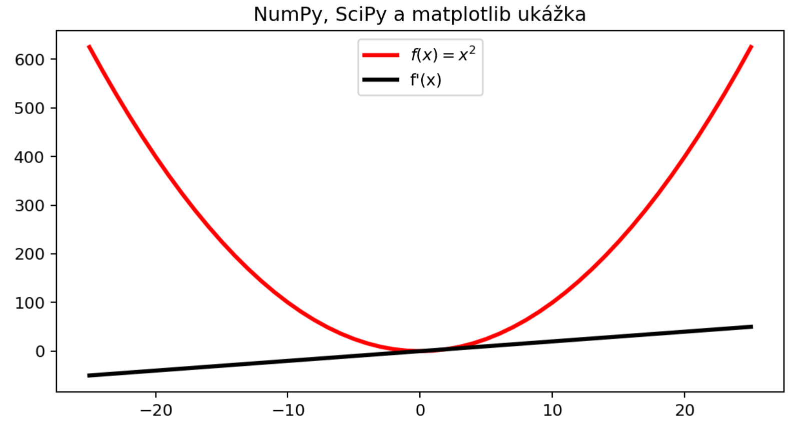
SciPy. Špecifickým dôvodom na použitie knižnice NumPy je aj využitie ďalšej knižnice, SciPy, ktorá je na NumPy priamo postavená, ako uvádza jej oficiálna dokumentácia [15]. SciPy, alebo aj Scientific Python, je knižnica, alebo aj kolekcia matematických algoritmov a určených funkcií. Pre potreby tejto práce poskytuje algoritmy na výpočet derivácií, či nulových bodov. S knižnicou SciPy sa Python a prostredie Jupyter Notebook stávajú plnohodnotným konkurentom spomínaných riešení ako MATLAB, The Interactive Data Language, či Octave.

matplotlib. Jednou z najpodstatnejších funkcií tejto bakalárskej práce je vizuálna interakcia s užívateľom. Funkcie a jej vlastnosti sa vizualizujú užívateľovi v podobe grafu a v rámci možností v čo najlepšej kvalite. Takéto vizualizačné funkcionality dodáva knižnica matplotlib. matplotlib je Python knižnica pre vytváranie a vizualizáciu najmä 2D grafov v produkčnej kvalite. Podporuje interaktívnu, aj neinteraktívnu vizualizáciu a umožňuje ukladať tieto vizualizácie vo forme obrázkov rozličných formátov (vektorových aj rastrových) typu JPG, PNG, PDF, PS a iných [11, 17].

Ukážka. Nasledovný kód demonštruje prácu s knižnicami NumPy, SciPy a matplotlib. V kóde sa zadefinuje funkcia $f(x) = x^2$ a uniformné delenie základného intervalu v podobe premennej `X`. Funkčné hodnoty (`Y`) a derivácie (`primes`) na celom intervale sa vypočítajú pomocou vektorizovaných funkcií knižníc NumPy, resp. SciPy. Cez modul pyplot knižnice matplotlib sa vypočítané hodnoty vykreslia do podoby grafu. (obr. 2.2).

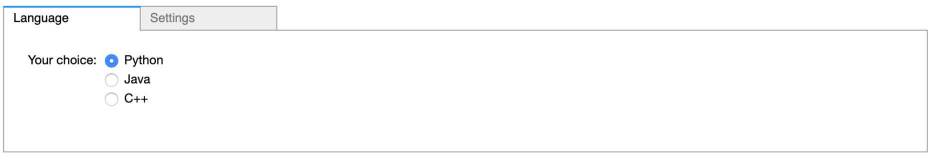
```
In [1]: %matplotlib notebook
        import matplotlib.pyplot as plt
        import numpy as np
        from scipy.misc import derivative

        def f(X): return X ** 2

        X = np.arange(-25, 25+1)
        Y, primes = f(X), derivative(f, X)
        fig, ax = plt.subplots()
        fig.set_size_inches(8, 4)
        ax.plot(X, Y, c='red', linewidth=2.5, label=r"$f(x) = x^2$")
        ax.plot(X, primes, c='black', linewidth=2.5, label=r"f'(x)")
        ax.set_title('NumPy, SciPy a matplotlib ukazka')
        ax.legend()
        fig.show()
```

ipywidgets. Editor ako softvér umožňuje interakciu užívateľa so zvyšnými časťami programu prostredníctvom myši, rôznych tlačidiel, textových vstupov, či iných ovládacích prvkov. Editor funguje striktne v prostredí Jupyter Notebook, a preto sú možnosti prispôsobenia



Obr. 2.2: Demonštrácia použitých knižníc - výstup

grafického užívateľského prostredia obmedzené. Potrebná časť upraviteľných ovládacích prvkov pre Jupyter Notebook je však dostupná pomocou rozšírenia ipywidgets. Ide o sadu HTML elementov ako tlačidlá, textové vstupy, checkboxy, obrázky, či animácie. Tieto elementy sa môžu združovať v rôznych rozloženiach ako okná, taby, či mriežky. V bunke Jupyter Notebooku je tak možné vytvoriť a spustiť vlastnú grafickú interaktívnu aplikáciu (obr. 2.3).



Obr. 2.3: Príklad interaktívnej aplikácie v prostredí Jupyter Notebook

Anaconda. Dostupnosť a manažment všetkých potrebných knižníc pre túto prácu, vrátane distribúcie samotného jazyka Python a prostredia Jupyter Notebook, zariaďuje programový balík Anaconda. Programový balík Anaconda je zadarmo a po nainštalovaní slúži okrem už uvedeného aj na spravovanie virtuálnych prostredí, poskytuje programy a nástroje pre distribúciu programovacieho jazyka pre štatistiku R a obsahuje vyše 7500 ďalších open source riešení najmä pre dátovú vedu [1].

2.2 Teoretické východiská

Pri tvorbe tejto bakalárskej práce bolo potrebné prispôsobiť zložitosť funkcionalít úrovni kurzu *Matematická analýza I*, v ktorom je uvažované využívanie jej programovej časti. Od toho sa odvíja úroveň matematickej teórie potrebnej k pochopeniu výpočtov na pozadí práce. Kurz v rámci vyšetrovania priebehu elementárnych funkcií postupne prechádza od definície reálnej funkcie o reálnej premennej, cez definíciu monotónnosti, derivácie, extrémov, až po konvexnosť, konkávnosť a inflexné body. Je potrebné presne zadefinovať tieto, ako aj ďalšie pojmy, poprípade doplniť pojmy, ktoré sa v osnovách predmetu priamo nenachádzajú, ako napr. metódy numerických výpočtov derivácií, či nulových bodov funkcie. Všetky tieto pojmy sú v práci využívané, či už priamo v rámci definovania funkcií, alebo na zabezpečenie algoritmickej funkcionality.

Väčšina matematických definícií a viet pochádza z vysokoškolských skrípt *Cvičenia z matematickej analýzy I* od autorov Kubáček a Valášek [8]. Základné pojmy týkajúce sa teórie chýb, či numerických výpočtov nulových bodov funkcie sú definované podľa knihy J. Eliáša [5] a samotné metódy a ich princípy sú vysvetlené podľa knihy K. Atkinsona [2]. Metódy výpočtu numerických derivácií sú zase popísané v citovaných učebných materiáloch [10, 12].

V tejto sekcii sa čitateľ stretne s niekoľkými definíciami a vetami, ktoré sú z časti napísané symbolickou notáciou. Aby boli tieto symboly čitateľovi jasné, nasledovné tabuľky ich vysvetľujú.

Logická a výroková notácia

$\varphi \wedge \psi$	platí φ a súčasne platí ψ
$\varphi \vee \psi$	platí φ alebo platí ψ
$\varphi \Rightarrow \psi$	ak platí φ, tak platí ψ
$\varphi \Leftrightarrow \psi$	φ platí práve vtedy, keď platí ψ
$\forall x \in A\colon \varphi(x)$	pre každý prvok x z množiny A platí $\varphi(x)$

Množinová notácia

$[x,y]$	usporiadaná dvojica prvkov x a y
$\{x \in A\ ;\ \varphi(x)\}$	množina všetkých $x \in A$, pre ktoré platí $\varphi(x)$
$\mathbb{R}$	množina reálnych čísel
$\mathbb{R} \times \mathbb{R}$	karteziánsky súčin množiny reálnych čísel
$A \subset B$	A je podmnožinou B
$A \cap B$	prienik množín A a B
$A \cup B$	zjednotenie množín A a B
$A \setminus B$	rozdiel množín A a B

V tabuľkách sú uvedené len také symboly a príslušné vysvetlenia, ktoré sa nachádzajú v tomto texte. Ďalšie symboly a notácie sú vysvetlené v literatúre od autorov Kubáček a Valášek [8], či v prehľade označení, operácií a skratiek v knihe *Úvod do inteligentního kalkulu* [3].

2.2.1 Funkcie

V tejto časti sú okrem pojmov funkcia, funkčná hodnota, definičný obor funkcie a graf funkcie definované aj elementárne funkcie. Uvedené sú jednotlivé druhy základných elementárnych funkcií, s ktorými sa môže študent stretnúť na kurze *Matematická analýza I* a ktoré slúžili ako predloha, či vstupy pri tvorbe tejto bakalárskej práce.

Funkcie. Nech $A \subset \mathbb{R}$ je množina. Ak je každému číslu $x \in A$ priradené práve jedno číslo $y \in \mathbb{R}$, ktoré označíme $f(x)$, hovoríme, že f je funkcia definovaná na množine A. Číslo $f(x)$ sa nazýva funkčná hodnota v bode x a množina A definičný obor funkcie f, označujeme ho aj ako $D(f)$. Pre zdôraznenie, že definičným oborom funkcie f je množina A, používame zápis $f : A \to \mathbb{R}$; prípadne $f : A \to B$, ak pre každé $x \in A$ platí $f(x) \in B$.

Množina

$$\{[x, f(x)] \in \mathbb{R} \times \mathbb{R}\,;\, x \in A\}$$

bodov roviny, kde $f : A \to \mathbb{R}$ je daná funkcia, sa nazýva graf funkcie f, pričom $[x, f(x)]$ je zápis bodu roviny pomocou jeho súradníc v danej súradnicovej sústave.

Elementárne funkcie. Funkcie, ktoré vzniknú zo základných elementárnych funkcií len použitím operácií súčtu, rozdielu, súčinu, podielu a superpozície funkcií, sa nazývajú elementárne funkcie. Medzi základné elementárne funkcie patria funkcie konštantné, mocninové, exponenciálne, logaritmické, goniometrické a cyklometrické.

Poznámka. Používané pojmy z matematickej analýzy ako intervaly, (rýdze) okolie bodu, spojitosť funkcie, vnútorný bod množiny, či limita funkcie sú považované za známe, avšak ich definície a súvisiace notácie sú k dohľadaniu v literatúre od Kubáčka a Valáška [8].

Derivácia funkcie. Pojem derivácie funkcie zastupuje kľúčovú funkcionalitu v rámci tejto bakalárskej práce. Okrem toho, že patrí do osnov predmetu *Matematická analýza I*, je derivácia využívaná na výpočet kľúčových hodnôt pri vyšetrovaní vlastností funkcie pomocou diferenciálneho počtu.

Nech x je vnútorný bod definičného oboru funkcie f. Hovoríme, že funkcia f má v tomto bode deriváciu $f'(x)$, ak

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

Tu predpokladáme, že limita na pravej strane existuje a je konečná.

Nech M je množina všetkých bodov definičného oboru $D(f)$, v ktorých má funkcia f deriváciu. Funkcia $f' : M \to \mathbb{R}$, ktorá každému bodu $x \in M$ priradí hodnotu $f'(x)$ derivácie funkcie f v bode x, sa nazýva derivácia funkcie f.

Derivácie vyšších rádov. V nasledujúcej sekcii 2.2.2 je uvedených niekoľko viet, ktoré slúžili ako podklad pre tvorbu algoritmov na vyšetrovanie niektorých vlastností funkcie pomocou diferenciálneho počtu. V niektorých týchto vetách sa používa pojem derivácie vyššieho rádu, preto je dôležité si ho korektne zadefinovať.

Nech x je vnútorný bod $D(f)$. Definície vyšších rádov definujeme induktívne: Nech funkcia f je definovaná v okolí bodu $x \in \mathbb{R}$, označme $f^{(0)} := f$. Hovoríme, že funkcia f má n-tú deriváciu v bode x, označujeme $f^{(n)}(x)$, ak existuje derivácia funkcie $f^{(n-1)}$ v bode x.

Derivácia funkcie $f^{(n-1)}$ sa nazýva n-tá derivácia funkcie f a označuje sa $f^{(n)}$. Okrem označení $f^{(1)}, f^{(2)}, f^{(3)}, f^{(4)}, f^{(5)}$, ... sa používajú aj označenia f', f'', f''', f'''' (alebo f^{IV}), f^{V}, ...

2.2.2 Vyšetrovanie priebehu funkcie

Extrémy. Hovoríme, že funkcia f má v bode $c \in D(f)$ ostré lokálne maximum, ak existuje okolie $O(c)$ bodu c také, že platí

$$\forall x \in (O(c) \setminus \{c\}) \cap D(f)\colon f(x) < f(c).$$

Ďalej hovoríme, že funkcia f má v bode $c \in D(f)$ ostré lokálne maximum, ak existuje okolie $O(c)$ bodu c také, že platí

$$\forall x \in (O(c) \setminus \{c\}) \cap D(f)\colon f(x) > f(c).$$

Ostré lokálne maximá a ostré lokálne minimá sa súhrnne nazývajú ostré lokálne extrémy.

Veta 2. *Nech funkcia f je dvakrát diferencovateľná vo vnútornom bode x množiny $D(f)$.*

1. *Ak $f'(x) = 0$ a $f''(x) > 0$, tak f má v bode x ostré lokálne minimum.*

2. *Ak $f'(x) = 0$ a $f''(x) < 0$, tak f má v bode x ostré lokálne maximum.*

Veta 3. *Nech funkcia f je n-krát $(n \geq 2)$ diferencovateľná vo vnútornom bode x množiny $D(f)$, nech $f'(x) = ... = f^{(n-1)}(x) = 0$ a $f^{(n)}(x) \neq 0$.*

1. *Ak n je párne a $f^{(n)}(x) > 0$, tak funkcia f má v bode x ostré lokálne minimum (ostré lokálne maximum).*

2. *Ak n je párne a $f^{(n)}(x) < 0$, tak funkcia f má v bode x ostré lokálne maximum.*

3. *Ak n je nepárne, funkcia f nemá v bode x lokálny extrém.*

Monotónnosť funkcie. Nech $A, B \subset \mathbb{R}$ sú množiny, kde $B \subset A$. Funkcia $f: A \to \mathbb{R}$ sa nazýva rastúca na množine B, ak platí

$$\forall x, y \in B\colon x < y \Rightarrow f(x) < f(y).$$

Funkcia $f: A \to \mathbb{R}$ sa nazýva klesajúca na množine B, ak platí

$$\forall x, y \in B\colon x < y \Rightarrow f(x) > f(y).$$

Funkcia, ktorá je rastúca na množine B alebo klesajúca na množine B sa nazýva rýdzomonotónna na množine B. Funkcia rastúca (klesajúca, rýdzomonotónna) na svojom definičnom obore sa nazýva rastúca (klesajúca, rýdzomonotónna).

Veta 4. *Nech funkcia $f: I \to \mathbb{R}$ je spojitá na intervale I a má deriváciu v každom jeho vnútornom bode. Potom*

1. *ak pre každý vnútorný bod x z intervalu I platí, že $f'(x) > 0$, tak f je rastúca na I;*
2. *ak pre každý vnútorný bod x z intervalu I platí, že $f'(x) < 0$, tak f je klesajúca na I.*

Inflexné body. Nech x je vnútorný bod $D(f)$. Bod x sa nazýva inflexný bod funkcie f, ak má v bode x deriváciu a existuje $\varepsilon > 0$ tak, že funkcia f je rýdzo konvexná na jednej z množín $(x-\varepsilon, x\rangle$, $\langle x, x+\varepsilon)$ a rýdzo konkávna na druhej z nich.

Veta 5. *Nech funkcia f je trikrát diferencovateľná v bode x a dvakrát diferencovateľná v niektorom jeho okolí. Ak platí $f''(x) = 0$ a $f'''(x) \neq 0$, tak x je inflexný bod funkcie f.*

Konvexnosť a konkávnosť. Funkcia f sa nazýva rýdzo konvexná na intervale $I \subset D(f)$, ak platí

$$\forall x, y \in I,\ x \neq y\ \forall p, q > 0,\ p+q = 1\colon f(px+qy) < pf(x) + qf(y).$$

Funkcia f sa nazýva rýdzo konkávna na intervale $I \subset D(f)$, ak platí

$$\forall x, y \in I,\ x \neq y\ \forall p, q > 0,\ p+q = 1\colon f(px+qy) > pf(x) + qf(y).$$

Veta 6. *Nech funkcia $f: I \to \mathbb{R}$ je spojitá na intervale I a dvakrát diferencovateľná v každom jeho vnútornom bode. Potom*

1. *ak pre každý vnútorný bod x z intervalu I platí, že $f''(x) > 0$, tak f je rýdzo konvexná na I;*
2. *ak pre každý vnútorný bod x z intervalu I platí, že $f''(x) < 0$, tak f je rýdzo konkávna na I.*

2.2.3 Teória chýb

Programová časť tejto práce implementuje niekoľko numerických metód, či algoritmov, ktorých výsledkami sú nejaké vypočítané hodnoty. Pri skúmaní týchto hodnôt je však vo väčšine prípadov potrebné rátať s nejakou odchýlkou, resp. chybou, ktorá vznikla pri hľadaní numerického riešenia, a teda považovať výslednú hodnotu len za "približnú", resp. "čo najpresnejšiu". Táto časť uvádza niekoľko pojmov z teórie chýb, ktoré definujú spôsob uvažovania pri skúmaní nameraných hodnôt z numerických výpočtov programovej časti tejto práce.

Približná hodnota čísla. Približnou hodnotou čísla X sa nazýva číslo x, ktoré sa málo líši od čísla X. Namiesto označenia približná hodnota čísla X môžeme použiť označenie aproximácia čísla X. Ak vieme, že pre približnú hodnotu x čísla X platí $x < X$, hovoríme, že číslo x je približnou hodnotou čísla X zdola (dolná aproximácia čísla X). Ak $x > X$, hovoríme, že číslo x je približnou hodnotou čísla X zhora (horná aproximácia čísla X).

Chyba približnej hodnoty čísla. Nech číslo x je približnou hodnotou čísla X. Chybou približnej hodnoty x čísla X, ktorú označujeme δx, nazývame číslo $X - x$, teda platí $\delta x = X - x$.

Absolútna chyba približnej hodnoty čísla. Absolútnou chybou približnej hodnoty x čísla X nazývame číslo $|\delta x| = |X - x|$.

Odhad absolútnej chyby približnej hodnoty čísla. Odhad absolútnej chyby približnej hodnoty x čísla X nazývame každé nezáporné číslo, ktoré nie je menšie ako $|\delta x|$ a označujeme ho Δx. Platí teda:

$$|X - x| \leq \Delta x$$

Relatívna chyba približnej hodnoty čísla. Relatívnou chybou ε približnej hodnoty x čísla X nazývame podiel absolútnej chyby Δx a absolútnej hodnoty samotnej približnej hodnoty $|x|$, ak $x \neq 0$. Platí teda:

$$\varepsilon x = \frac{\Delta x}{|x|} \qquad x \neq 0$$

Zápis približných hodnôt čísiel v desiatkovej číselnej sústave. V desiatkovej číselnej sústave každé $x \in \mathbb{R}$ môžeme zapísať v tvare

$$x = \pm(\chi_m 10^m + \chi_{m-1} 10^{m-1} + \ldots + \chi_{m-n+1} 10^{m-n+1} + \ldots),$$

kde χ_i sú číslice čísla x, $\chi_i = 0, 1, \ldots 9$, pričom $\chi_m \neq 0$ a m je celé číslo, ktoré nazývame rádom čísla x.

Hodnotné číslice. Približné hodnoty čísiel sa v praxi najčastejšie vyskytujú vo forme konečného desatinného rozvoja,

$$x = \pm(\chi_m 10^m + \chi_{m-1} 10^{m-1} + \ldots + \chi_{m-n+1} 10^{m-n+1}) \qquad \chi_m \neq 0.$$

Hodnotnou číslicou približnej hodnoty x čísla X vyjadreného v desatinnom rozvoji potom nazývame:

(a) všetky nenulové číslice,

(b) všetky nuly medzi prvou a poslednou nenulovou číslicou,

(c) všetky také nuly vpravo od poslednej nenulovej číslice, ktoré sa používajú na označenie počtu jednotiek rádov, ktoré v približnej hodnote x čísla X nechávame.

Platné číslice. Hovoríme, že prvých n hodnotných číslic približnej hodnoty x čísla X je platných, keď absolútna chyba približnej hodnoty x nie je väčšia ako polovica jednotky rádu rovného rádu n-tej hodnotnej číslice, ak počítame zľava doprava.

Ak pre približnú hodnotu x čísla X, zapísanú v desiatkovej číselnej sústave, platí

$$\Delta x = |X - x| \leq \frac{1}{2} 10^{m-n+1},$$

potom podľa definície platných číslic sú číslice $\chi_m, \chi_{m-1}, \ldots, \chi_{m-n+1}$ platnými číslicami približnej hodnoty x.

2.2.4 Numerický výpočet derivácií

Aby mohli byť vety zo sekcie 2.2.2 implementované vo výstupnom programe tejto práce, potrebuje mať program k dispozícii predpisy prvých derivácií, či derivácií vyšších rádov. Tie môžu byť v prípade jednoduchších predpisov funkcií odvodené analyticky, čo však nie je vždy pravidlom, keďže analytický výpočet derivácií môže byť príliš náročný, dokonca nemožný. Program tak musí využiť informácie o funkčných hodnotách danej funkcie na nejakej množine bodov a pokúsiť sa dopočítať, resp. aproximovať derivácie numericky. Algoritmus pre numerické aproximovanie derivácií, ktorý je používaný vo výstupnom programe tejto práce, vychádza z niekoľkých pojmov a postupov, ktoré sú popísané práve v tejto časti.

Taylorov rad. Mnoho známych vzorcov pre aproximáciu numerických derivácií, vrátane metódy centrálnej diferencie, vyplýva z rozvoja Taylorovho radu.

Nech funkcia f je n-krát diferencovateľná v bode $a \in \mathbb{R}$. Potom mocninový rad

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

sa nazýva Taylorov rad funkcie f v bode a.

Dopredná diferencia. Jednoduchá aproximácia prvej derivácie vychádza priamo z definície derivácie

$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$

kde uvažujeme $h > 0$. Táto aproximácia prvej derivácie funkcie sa nazýva metóda doprednej diferencie. Táto metóda je napríklad v prípade lineárnych funkcií presným vzorcom pre výpočet prvej derivácie, no takmer pri všetkých ostatných druhoch funkcií tomu tak nie je.

Odvodenie tejto metódy a jej vzťahu k presnosti sa dá ukázať využitím rozvoja Taylorovho radu a položením $(x-a) = h$ pre konzistentnosť zápisu:

$$f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2} + f'''(x)\frac{h^3}{3!} + \cdots .$$

Následnými úpravami tohto rozvoja môžeme uviesť presný vzorec pre aproximáciu prvej derivácie funkcie, a teda

$$f'(x) = \frac{f(x+h) - f(x)}{h} - f''(x)\frac{h}{2} - f'''(x)\frac{h^2}{3!} - \cdots .$$

Prvý člen na pravej strane rovnice je odvodená dopredná diferencia, pričom zvyšné členy môžeme nahradiť označením $O(h)$, kde exponent α pri h v $O(h^\alpha)$ vyjadruje stupeň presnosti tejto metódy

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h).$$

Túto presnosť odvodzujeme vždy od prvého člena rozvoja, ktorý sme zanedbali, s implikáciou, že zvyšné členy väčších stupňov sú s $h \to 0$ vždy menšie ako tento prvý člen. V prípade vyššie odvodenej metódy doprednej diferencie tak môžeme povedať, že má presnosť prvého stupňa.

Spätná diferencia. Aproximácia prvej derivácie v bode x pomocou metódy doprednej diferencie je založená na hodnotách funkcie v bodoch x a $x+h$. Ak pre aproximáciu použijeme hodnoty funkcie v bodoch $x-h$ a x, tak obdobným spôsobom odvodíme vzťah

$$f'(x) = \frac{f(x) - f(x-h)}{h} + O(h),$$

ktorý nazývame spätná diferencia a rovnako ako dopredná diferencia má presnosť prvého stupňa.

Centrálna diferencia. Pomocou Taylorovho radu a ďalších bodov v okolí bodu x vieme odvodiť aj vzorce s vyšším stupňom presnosti. Rozdielom dvoch Taylorovych radov

$$f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2} + f'''(x)\frac{h^3}{3!} + \cdots$$
$$f(x-h) = f(x) - f'(x)h + f''(x)\frac{h^2}{2} - f'''(x)\frac{h^3}{3!} + \cdots$$

dostaneme napríklad vzťah pre výpočet prvej derivácie funkcie f v bode x

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2),$$

ktorý nazývame centrálna diferencia, pričom presnosť sa zvýšila na stupeň 2. Ešte vyššiu presnosť môžeme dosiahnuť pridaním ďalších bodov do okolia bodu x (zjemnenie intervalu). Napríklad vzorec s presnosťou stupňa 4 má tvar

$$f'(x) = \frac{f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)}{12h} + O(h^4).$$

Centrálna diferencia je najčastejšie využívanou metódou pre výpočet derivácií v tejto práci, a to aj v súvislosti s výpočtom derivácií vyšších rádov. Napríklad sčítaním vyššie uvedených dvoch rozvojov Taylorovho radu sa členy $f'(x)h$ navzájom odstránia a po malých úpravách dostaneme vzťah pre výpočet druhej derivácie funkcie f v bode x s presnosťou stupňa 2:

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2)$$

Presnosť metód doprednej, spätnej a centrálnej diferencie na aproximáciu prvej derivácie funkcie $f(x) = x^3 + 3x^2 - 9x - 10$ v danom bode x ilustruje obrázok 2.4.

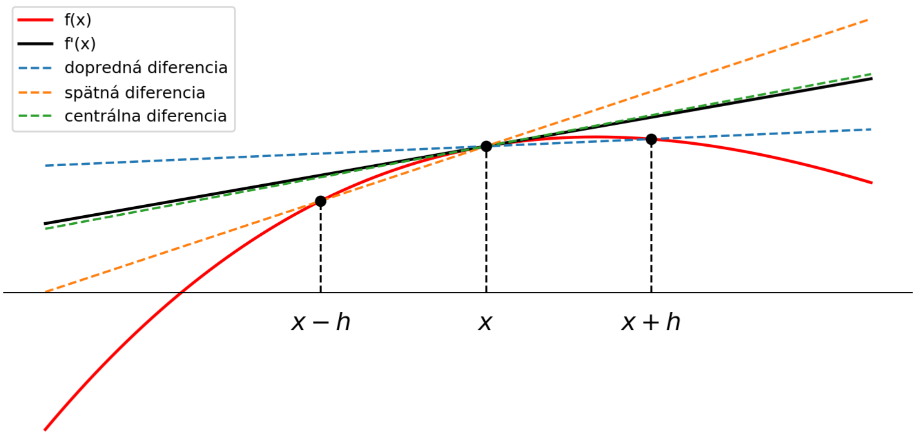
2.2.5 Hľadanie nulových bodov funkcie

Súčasťou práce sú aj dve numerické metódy na aproximáciu nulových bodov funkcie. Metódy sa rozlišujú podľa toho, či sa využije prvá derivácia funkcie, a to nasledovne:

- ak sa prvá derivácia funkcie využíva, nulové body funkcie sú aproximované *Newtonovou metódou*
- ak sa derivácia funkcie nevyužíva, nulové body funkcie sú aproximované *metódou sečníc*

Nasledujúce odseky vysvetľujú základné pojmy dôležité pri hľadaní nulových bodov funkcie, resp. koreňov rovnice a princípy spomenutých metód ich hľadania (aproximovania).

Základné pojmy. Nech $f(x) = 0$ je rovnica, kde $f(x)$ je funkcia definovaná na množine M. Číslo α nazývame koreňom alebo riešením rovnice $f(x) = 0$, keď platí $f(\alpha) = 0$. Riešiť



Obr. 2.4: Numerická aproximácia prvej derivácie

rovnicu $f(x) = 0$ znamená nájsť všetky jej riešenia. Ak $f(x)$ je polynóm n-tého stupňa, t.j. ak rovnica $f(x) = 0$ má tvar

$$a_0 x^n + a_1 x^{n-1} + \ldots + a_n = 0 \qquad a_0 \neq 0$$

a nazývame ju algebraickou rovnicou n-tého stupňa o jednej neznámej. Rovnicu, ktorá nie je algebraická, nazývame nealgebraickou rovnicou.

Približné alebo numerické riešenie rovnice $f(x) = 0$ spočíva vo všeobecnosti v zostrojení postupnosti približných hodnôt koreňa α rovnice $f(x) = 0$

$$x_1, x_2, \ldots, x_n, \ldots,$$

ktorá konverguje, t.j. existuje

$$\lim_{n\to\infty} x_n = \varphi; \qquad f(\varphi) = 0.$$

n-tý člen postupnosti $x_1, x_2, \ldots, x_n, \ldots$ nazývame n-tou približnou hodnotou koreňa φ alebo n-tou aproximáciou koreňa φ rovnice $f(x) = 0$. Číslo n hľadáme tak, aby absolútna chyba približnej hodnoty x_n koreňa φ rovnice $f(x) = 0$, t.j. $|x_n - \varphi|$ bola dostatočne malá, alebo aby bola menšia ako vopred dané kladné číslo ε.

Separáciou koreňov rovnice $f(x) = 0$ rozumieme hľadanie takých intervalov na číselnej osi, v ktorých leží práve jeden koreň rovnice $f(x) = 0$. Predpokladajme, že sme našli interval, v ktorom leží jediný koreň rovnice $f(x) = 0$. Spôsob hľadania približnej hodnoty x_n koreňa φ rovnice $f(x) = 0$ v tomto intervale tak, aby táto približná hodnota aproximovala koreň φ s vopred prepísanou absolútnou chybou ε, nazývame aproximácia koreňa rovnice $f(x) = 0$.

Newtonova metóda. Newtonova metóda, inak známa aj ako metóda dotyčníc, je podľa Atkinsona [2] najznámejšou procedúrou pre nájdenie koreňov rovnice pre jej formálnu jednoduchosť a rýchlosť, no nie je vždy najvhodnejšou metódou na riešenie každého nelineárneho problému.

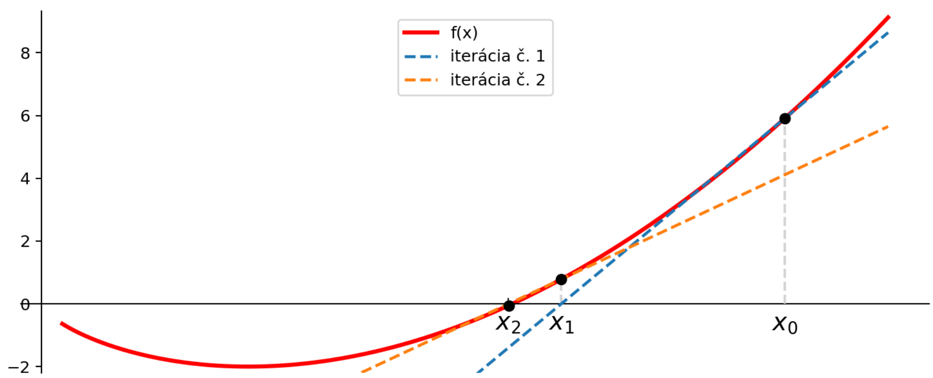
Nech x_0 je počiatočný odhad pre koreň α rovnice $f(x) = 0$. Newtonová metóda má za úlohu nájsť postupnosť približných hodnôt koreňa α rovnice $f(x) = 0$, ktorá by mala postupne konvergovať k bodu α. Hľadanie tejto postupnosti prebieha nasledovne: pri predpoklade, že x_0 je dostatočne blízko α, metóda aproximuje graf funkcie $y = f(x)$ v okolí bodu α, a to tak, že v bode $[x_0, f(x_0)]$ skonštruuje dotyčnicu ku grafu funkcie. Koreňom tejto dotyčnice je potom nová aproximácia α, označená x_1. Postupným opakovaním (iterovaním) tohto postupu všeobecne pre

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

sa metóda dopracuje k spomínanej postupnosti približných hodnôt koreňa α rovnice $f(x) = 0$. Postup prvých dvoch iterácií na príklade funkcie

$$f(x) = x^2 - 3\sqrt[3]{x^2}$$

je znázornený obrázkom 2.5 s aproximovanými bodmi x_1 a x_2. Tabuľka 2.1 zase ukazuje namerané hodnoty pri rovnakej funkcii, a to pri štyroch iteráciách.



Obr. 2.5: Dve iterácie Newtonovej metódy

Štvrtý stĺpec tabuľky 2.1 znázorňuje, ako sa s každou iteráciou Newtonovej metódy približné hodnoty x_n koreňa α približujú k hodnote α, dopočítaného programom WolframAlpha (`www.wolframalpha.com`). Posledný stĺpec zase znázorňuje postupné zmenšovanie odchýlky jednotlivých približných hodnôt x_n koreňa α.

Newtonova metóda na príklade konverguje naozaj veľmi rýchlo, a to hneď, ako sa pri-

n	x_n	$f(x_n)$	$\alpha - x_n$	$x_{n+1} - x_n$
0	3.6	5.91323825	-1.32049294	-1.082999
1	2.517001	0.78422221	-0.23749394	-0.2551462
2	2.2618548	-0.05330492	0.01765226	0.0213885
3	2.2832433	0.01137122	-0.00373624	-0.00447446
4	2.27876884	-0.00224309	0.00073822	

Tabuľka 2.1: Vypočítané hodnoty štyroch iterácií Newtonovej metódy

bližná hodnota x_n koreňa α blíži ku hodnote α. Väčšie odchýlky pri prvej, resp. druhej iterácií sú dôsledkom zvolenia nie príliš presného počiatočného odhadu bodu x_0, t.j. aproximácie koreňa rovnice. Z toho vyplýva, že zlepšením aproximácie koreňa rovnice je možné ešte viac urýchliť konvergenciu Newtonovej metódy.

Metóda sečníc. Predpis, či hodnoty prvej derivácie funkcie môžu byť z rôznych dôvodov nedostupné. V takom prípade je použitie Newtonovej metódy a aproximovanie grafu funkcie pomocou dotyčníc v jednom bode nemožné. Riešením je použitie metódy sečníc, ktorá funguje veľmi podobne ako Newtonova metóda.

Nech x_0 a x_1 sú počiatočné odhady pre koreň α rovnice $f(x) = 0$. Metóda sečníc má za úlohu nájsť postupnosť približných hodnôt koreňa α rovnice $f(x) = 0$, ktorá by mala postupne konvergovať k bodu α. Hľadanie tejto postupnosti prebieha nasledovne: pri predpoklade, že x_0 a x_1 sú dostatočne blízko α, metóda aproximuje graf funkcie $y = f(x)$ v okolí bodu α, a to tak, že pomocou bodov $[(x_0, f(x_0)]$ a $[x_1, f(x_1)]$ skonštruuje sečnicu grafu funkcie. Koreňom tejto sečnice je potom nová približná hodnota x_2 koreňa α. Výpočet vychádza z rovnice sečnice, ktorá má tvar

$$y = \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_1) + f(x_1),$$

pričom koreňom rovnice tejto sečnice, resp. novou približnou hodnotou α je spomínaný bod

$$x_2 = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}.$$

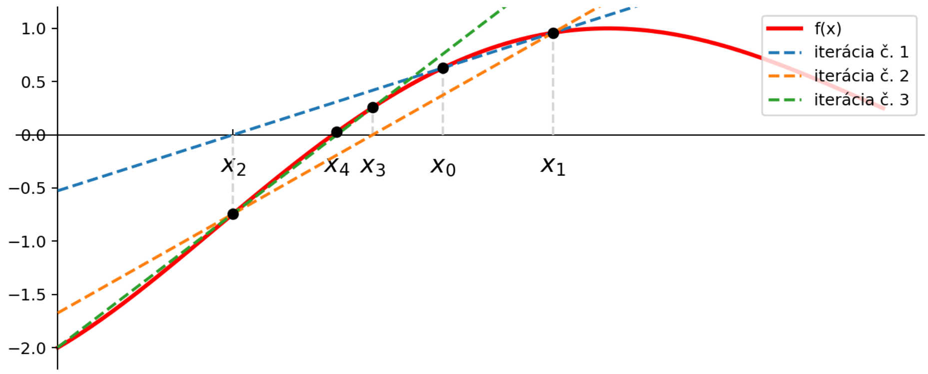
Postupným opakovaním (iterovaním) tohto postupu všeobecne pre

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \qquad n \geq 1$$

sa metóda dopracuje k spomínanej postupnosti približných hodnôt koreňa α rovnice $f(x) = 0$. Postup prvých troch iterácií na príklade funkcie

$$f(x) = 2x^4 - 7x^3 + 5x^2 + 3x - 2$$

je znázornený obrázkom 2.6 s nájdenými približnými hodnotami x_2, x_3 a x_4 koreňa α. Štvrtý stĺpec tabuľky 2.2 zase znázorňuje, ako sa s každou iteráciou metódy sečníc približné hodnoty x_n separovaného koreňa $\alpha = \frac{1}{2}$ funkcie približujú práve k tejto hodnote.



Obr. 2.6: Tri iterácie metódy sečníc

n	x_n	$f(x_n)$	$\alpha - x_n$	$x_{n+1} - x_n$
0	0.7	0.6292	-0.2	0.2
1	0.9	0.9592	-0.4	-0.58133333
2	0.31866667	-0.74215433	0.18133333	0.25358565
3	0.57225232	0.25681816	-0.07225232	-0.06519239
4	0.50705994	0.0263491	-0.00705994	-0.00745332
5	0.49960661	-0.0014756	0.00039339	

Tabuľka 2.2: Vypočítané hodnoty piatich iterácií metódy sečníc

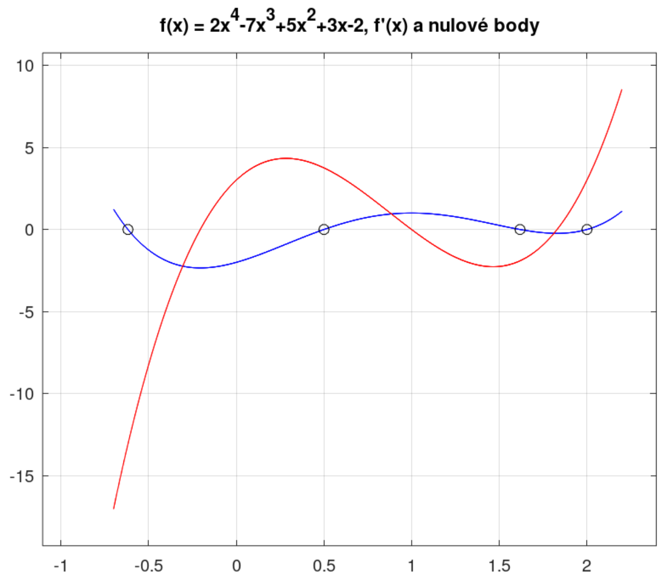
2.3 Existujúce riešenia

Editor, ako výstup tejto práce, využíva vety a definície zo sekcie 2.2. Zaobaľuje ich do formy funkcií, či algoritmov, ktoré riadia funkcionalitu programu. Tieto funkcionality, spolu s vizualizáciou grafov je možné v rôznych podobách implementovať aj v iných, verejne dostupných nástrojoch. V tejto sekcii je popísaný jeden komerčný a dva open-source nástroje, ktoré môžu slúžiť ako prípadná alternatíva pre študenta analyzujúceho priebeh elementárnych funkcií.

MATLAB. MATLAB je vysoko-úrovňový jazyk a interaktívne prostredie pre numerické výpočty, vizualizáciu a programovanie. Umožňuje analyzovať dáta, vyvíjať algoritmy, či vytvárať modely a aplikácie. Jazyk, jeho nástroje a zabudované matematické funkcie umožňujú zvoliť mnoho prístupov pri riešení problému a dosiahnuť toto riešenie rýchlejšie v porovnaní s hárkami tabuľkového procesora, či programovacími jazykmi C/C++ alebo Java. MATLAB je využívaný v mnohých aplikáciách ako spracovanie signálu, obrazu, či videa, kontrolné systémy, systémy pre testovanie a meranie, finančníctvo alebo biológia [4]. Používanie softvéru MATLAB vyžaduje platenú licenciu.

GNU Octave. GNU Octave je tiež vysoko-úrovňový jazyk a interaktívne prostredie pre numerické výpočty, pričom v zmysle účelu tejto práce poskytuje rovnaké možnosti ako MATLAB, no je distribuované pod GNU licenciou. Licencia GNU používateľovi umožňuje využívať tento softvér zadarmo v plnom rozsahu, šíriť ho a dokonca aj prípadne upravovať. Obidva jazyky sú numerické, nie symbolické a ich základným typom je matica, pričom softvér je plne optimalizovaný pre vektorové operácie [9]. Nasledovný príklad demonštruje hľadanie nulových bodov a derivácie polynomiálu $f(x) = 2x^4 - 7x^3 + 5x^2 + 3x - 2$ v prostredí GNU Octave, pričom kód je aplikovateľný aj v prostredíi MATLAB.

```
> x = [-0.7:0.001:2.2];
> p = [2, -7, 5, 3, -2];
> y = polyval(p, x);
> dydx = diff(y) ./ diff(x);
> r = roots(p);
  r = 2.00000, 1.61803, -0.61803, 0.50000
> plot(x, y, 'b', x(1:end-1), dydx, 'r', r, 0, 'ko')
> title("f(x) = 2x^4-7x^3+5x^2+3x-2, f'(x) a nulove body")
```



Obr. 2.7: Demonštrácia MATLAB/Octave - výstup

Práca s nástrojmi popísanými v tejto časti je najmä z hľadiska funkcionalít porovnateľných s touto bakalárskou prácou podobná a vzhľadom na ich robustnosť aj flexibilnejšia.

To však predurčuje užívateľa k väčšej znalosti jednotlivých jazykov a ovládacích procesov, čím sa komplikuje prístup k jednoduchým prvkom matematickej analýzy. Na druhej strane editor, ako predmet tejto bakalárskej práce, obsahuje nevyhnutné minimum funkcionalít a grafických výstupov potrebných k splneniu edukačných potrieb študenta, pričom mu k nim umožňuje pristupovať interaktívne, cez grafické rozhranie.

2.4 Podobné práce

Táto bakalárska práca sa dá považovať ako priama alternatíva pre diplomovú prácu Jakuba Trubača, *Vývoj interaktívnych dokumentov pre výučbu matematickej analýzy*, ktorá bola vypracovaná pod vedením toho istého školiteľa. Práca tak vychádza z pomerne rovnakých základov. Autor vytvoril a popísal rozšírenie funkcionalít interaktívneho prostredia Jupyter Notebook vo forme editora. Tento editor bol vytvorený za účelom efektívnejšej analýzy priebehu funkcií na cvičeniach k predmetu *Matematická analýza*.

Popis riešenia. Editor na vstupe spracuje užívateľom zadané hodnoty, vykreslí graf funkcie a umožní nájsť nulové body, extrémy funkcie, inflexné body, či s tým spojené vlastnosti ako monotónnosť, konkávnosť, alebo konvexnosť. Jedným z najdôležitejších parametrov je metóda na hľadanie nulových bodov. Autor zvolil Newtonovu metódu, ktorá iteratívnym procesom vyberie vhodného kandidáta na nulový bod. Na vstupe očakáva jeden štartový bod, akýsi počiatočný tip, ktorý si môže užívateľ interaktívne meniť a dopracovať sa tak k prípadným presnejším výsledkom. Z hľadiska používateľského rozhrania autor v jeho práci využíva knižnicu ipywidgets, ktorá má na starosti interaktívne ovládacie prvky v prostredí Jupyter Notebook. Po spracovaní užívateľských hodnôt a vykreslení grafu sa pod ním zobrazia ovládacie prvky v podobe posuvných tlačidiel, či označovacích boxov na výber príslušných hodnôt. Editor má viacero okien, medzi nimi okno na samotnú analýzu, okno na prispôsobenie Newtonovej metódy a okno na výstupné informácie.

Komparácia. Hlavný rozdiel medzi prácami je práve v prístupe k Newtonovej metóde, zatiaľ čo Trubač túto metódu naprogramoval, v tejto práci sa využíva metóda z už naprogramovanej knižnice scipy. To umožňuje využiť silu knižnice na numerickú matematiku numpy a prijať ako argumenty celé vektory hodnôt, nie len jediný bod - počiatočný tip. Práca s vypočítanými údajmi a ich spracovanie je tak odlišná vzhľadom na túto vlastnosť. Knižnica scipy rovnako umožní zanedbať zadanie prvej derivácie funkcie a použije pre výpočet nulových bodov metódu sečníc. Rozdiely v grafickom prevedení a ovládaní nie sú markantné. V tejto práci sa graf funkcie nachádza priamo v editore, čím tvorí jeho značnú časť a všetky hlavné ovládacie prvky sa nachádzajú vedľa neho. Doplnené sú taktiež výstupy, avšak aj v podobe zaznamenávania užívateľských krokov, čím je užívateľ sprevádzaný celou analýzou funkcie. Rovnako je vynechaná možnosť dodatočne špecifikovať interval analýzy, v tomto prípade si interval merania určí užívateľ jeho definovaním ešte pred samotným spustením editora.

3 Návrh riešenia

Výstupný program tejto práce, ako edukačný nástroj, má presne definované požiadavky vychádzajúce primárne zo skúseností vyučujúceho. Návrh tohto programu tak vychádza priamo z týchto požiadaviek, ktoré sa dajú zhrnúť do štyroch hlavných kategórií:

1. analýza funkcií pomocou numerických algoritmov;
2. grafické užívateľské rozhranie;
3. konfigurácia programu;
4. spúšťanie a manažovanie inštancií programu.

Tieto štyri kategórie sú inšpiráciou pre rozdelenie programu na štyri moduly, ktoré vo výsledku tvoria výstupný program tejto práce. Každý nasledujúci popis modulu obsahuje detailnejšie požiadavky na ich obsah a formu, popis uvažovaných tried a metód, ktoré zaručujú funkčnosť daného modulu a prípadný popis komunikácie s inými modulmi.

3.1 Analýza funkcií

Analýza priebehu funkcie je jadrom tejto bakalárskej práce, pričom sa opiera najmä o vhodne zvolené, či naprogramované algoritmy. Dôležité je aj zvolenie príslušných dátových štruktúr, v ktorých si uchováva, resp. pomocou ktorých presúva už vypočítané údaje. Medzi základné požiadavky týkajúce sa tohto modulu ďalej patria:

1. manažment parametrov užívateľom zadanej funkcie;
2. výpočet numerických derivácií cez modul `scipy.misc.derivative`;
3. výpočet nulových bodov funkcie cez modul `scipy.optimize.newton`;
4. výpočet vlastností priebehu funkcie pomocou vypočítaných derivácií;
5. vykresľovanie vypočítaných hodnôt do grafu prostredníctvom knižnice `matplotlib`.

Užívateľská funkcia. Klasickým scenárom pri spúšťaní programu je prvotné zadefinovanie funkcie, parametrov jej grafu a následné vykreslenie tohto grafu. Úlohou programu je zdediť okrem samotného predpisu funkcie aj čo najväčšie množstvo parametrov jej grafu a kopírovať tak v čo najväčšej možnej miere užívateľské preferencie.

Všetky tieto užívateľské parametre by mali byť sprístupnené a zdieľané v rámci jediného objektu užívateľskej funkcie. Na ukladanie parametrov sú uvažované také dátové štruktúry, ktoré ich umožnia dynamicky a efektívne meniť počas behu programu. Objekt užívateľskej funkcie tak obsahuje metódy `get_parameter` a `set_parameter`.

Okrem parametrov týkajúcich sa samotnej funkcie, či parametrov grafu, objekt užívateľskej funkcie združuje všetky vypočítané hodnoty v ďalšej dátovej štruktúre, pričom aj k nim je zabezpečený efektívny prístup a ich prípadná zmena.

Program je pripravený pracovať s viacerými objektami užívateľskej funkcie (analyzovať viacero funkcií v jednej inštancii), hoci to nie je podľa požiadaviek nutné. Nad týmito objektami je tak uvažovaná trieda, ktorá ich združuje v jedinej štruktúre a umožňuje medzi nimi vyberať. Plní tak riadiacu funkcionalitu, t.j. spúšťa prepočty dát týchto funkcií. Obsahuje aj nevyhnutné objekty týkajúce sa samotného grafu funkcie a zaručuje tak, že počas behu programu existuje jediné výstupné okno s vizualizáciou dát, ktoré sa mení len v prípade ich prepočtov, resp. po vyžiadaní od užívateľa.

Numerické výpočty. Zrejme najdôležitejšie údaje, ktoré poskytuje modul na analýzu funkcií, sú derivácie funkcie vypočítané numerickým spôsob za použitia externej knižnice SciPy. V alternatívnom prípade je program podľa požiadaviek schopný vypočítať derivácie z ich predpisov poskytnutých užívateľom. Program si vždy udržiava hodnoty minimálne prvých troch derivácií danej funkcie, či už vypočítané numericky, podľa užívateľských predpisov alebo kombináciou týchto dvoch spôsobov.

Pomocou metódy `scipy.misc.derivative` sa počítajú numerické derivácie na celom intervale (na vstupe akceptuje vektory hodnôt) prostredníctvom metódy centrálnej diferencie (2.2.4). V prípade užívateľom poskytnutých predpisov derivácií sa predpokladá ich `ufunc` verzia, t.j. sú schopné taktiež pracovať s vektormi hodnôt. Vypočítané derivácie sa ukladajú do analytického datasetu objektu užívateľskej funkcie a sú pripravené pre ďalšie algoritmy výpočtov priebehu funkcie, ktoré ich využívajú.

Nulové body sa tiež počítajú numerickým spôsobom za použitia metódy externej knižnice `scipy.optimize.newton`. Ako už napovedá jej názov, na výpočet nulových bodov sa používa primárne Newtonova metóda, využívajúca aproximáciu nulového bodov pomocou dotyčníc (2.2.5). Metóda `scipy.optimize.newton` však v sebe skrýva aj metódu sečníc (2.2.5), pričom použitie jednotlivých metód závisí práve od poskytnutia predpisov prvej derivácie užívateľom. Ak tomu tak je, použije sa Newtonova metóda a ak nie, použije sa metóda sečníc.

Výpočet vlastností priebehu funkcie. Po vypočítaní derivácií vo všetkých bodoch zadaného intervalu je program pripravený spúšťať algoritmy vyplývajúce z viet o vyšetrovaní priebehu funkcie pomocou diferenciálneho počtu (2.2.2). Pre každú z vlastností, ako ostré lokálne extrémy, intervaly monotónnosti, inflexné body, či intervaly rýdzej konvexnosti a rýdzej konkávnosti, je implementovaný vlastný algoritmus, ktorý analyzuje príslušné derivácie podľa spomínaných viet z časti 2.2.2 a poskytuje s určitou presnosťou výsledky. Operácie v týchto algoritmoch sú taktiež optimalizované pre vektory hodnôt za využitia funkcií a metód knižnice Numpy.

Vykresľovanie do grafu. Algoritmy pre výpočet derivácií, nulových bodov a ostatných vlastností priebehu funkcie poskytujú užívateľovi informáciu vizualizovateľnú vo výstupnom okne pre graf. Pre každý typ tejto informácie je v programe vytvorená samostatná vykresľovacia metóda, ktorá do objektu grafu tieto hodnoty pridá, pokiaľ si to užívateľ vyžiadal výberom v hlavnom menu programu (parameter `visible`). Všetky vykresľovacie metódy združuje kontajnerová trieda, ktorá pri jej inštancovaní získava predvolené parametre grafu a vypočítané údaje z dátových štruktúr objektu užívateľskej funkcie.

3.2 Grafické užívateľské rozhranie

Prevedenie zložitých algoritmov a ich výpočtov do interaktívnej vizuálnej formy je ďalšou dôležitou požiadavkou pre vývoj tohto programu. Ako už bolo prezentované v motivačnej kapitole (1), vyšetrovanie priebehu funkcie by malo byť v maximálnej možnej miere prívetivé a nenáročné, nevyžadujúc od užívateľa žiadne pokročilé programátorské (ale aj matematické) techniky. Požiadavky na tento modul sú nasledovné:

1. prehľadná pracovná plocha s dominantnou vizualizáciou grafu;
2. jednoduché interaktívne prvky v podobe tlačidiel, resp. textových polí;
3. prehľadné výstupné okná s informáciami o užívateľských krokoch a vypočítaných hodnotách;
4. jednoznačná ovládacia logika, kontrola rôznych kombinácií vstupov.

Pracovná plocha a výstupné okná. Práca s interaktívnymi grafickými prvkami je v prostredí Jupyter Notebook zabezpečená použitím knižnice ipywidgets. Grafický výstup programu tvorí jedno okno na celú šírku bunky prostredia Jupyter Notebook. Uvažované je rozdelenie tohto okna na tri časti podľa kontextu - analytické okno, okno s chronologickým prehľadom užívateľských krokov a vypočítaných hodnôt a okno s upozorneniami vygenerovanými počas výpočtov dát, tiež v chronologickom poradí.

Dominantami všetkých troch okien sú výstupné okná, v prípade analytického okna samotný graf a v prípade okien výstupov a upozornení okná pre textové výpisy. Okno pre analýzu funkcií je uvažované ako najčastejšie používané, tým pádom je aplikovaná požiadavka pre výpis niektorých krátkych informácií o behu programu už priamo v tomto okne. Pod samotným výstupným oknom pre graf funkcie je tak umiestnené sekundárne výstupné okno, ktoré užívateľovi zobrazuje podstatné informácie v skrátenej forme, bez potreby zmeny používaného okna.

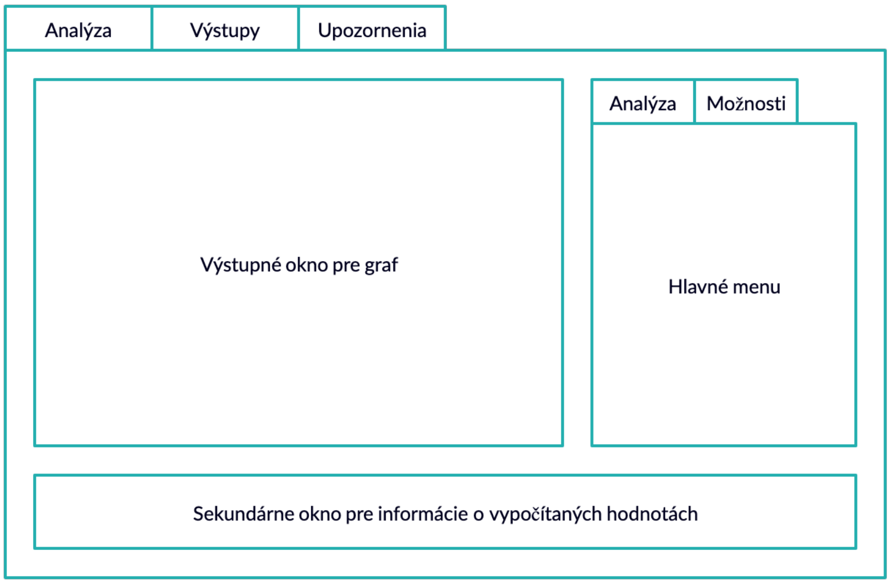
Hlavné menu. Ovládanie programu zabezpečujú ovládacie prvky umiestnené v hlavnom menu, ktoré je súčasťou analytického okna. Hlavné menu podľa požiadaviek obsahuje tlačidlá pre zapnutie, resp. vypnutie zobrazovania nulových bodov, ostrých lokálnych extrémov, intervalov, kde je funkcia rastúca, resp. klesajúca a intervalov, kde je funkcia rýdzo konvexná, resp. rýdzo konkávna. Uvažovanou funkcionalitou je aj zmena farby príslušných vlastností priebehu funkcie v grafe. To je docielené pridaním tlačidla pre zmenu farieb ku každému spomínanému tlačidlu pre zapnutie (vypnutie) požadovanej vlastnosti priebehu funkcie. Výber farby je však podmienený zapnutým tlačidlom danej vlastnosti, ak tomu tak nie je, výber farby je pre užívateľa zablokovaný.

Medzi ďalšie prvky hlavného menu patrí zmena počtu hodnôt intervalov na osi x (zjemnenie osi x). Na to je uvažovaný interaktívny objekt s možnosťou výberu celého čísla, ideálne reprezentovaného niekoľkými mocninami čísla 10. Toto číslo potom reprezentuje násobok pôvodného počtu prvkov daného intervalu.

Pre možnosť ovplyvnenia výsledkov metódy `scipy.optimize.newton` je do hlavného menu umiestnený interaktívny prvok s možnosť zadania počtu krokov (iterácií) tejto metódy. Podobne sa v hlavnom menu ráta s užívateľskou možnosťou spresnenia vypočítaných hodnôt, a to ich zaokrúhlením na zadaný počet platných cifier. Zadanie počtu krokov metódy `scipy.optimize.newton` aj počtu platných cifier je obmedzené v presných intervaloch s možnosťou ich zmeny v konfiguračnom súbore.

Uvažované je aj sekundárne menu, obsahujúce zmenu farby užívateľskej funkcie a tlačidlá pre zobrazenie, resp. zmenu farby prvých troch derivácií. Prítomné je aj tlačidlo na vykreslenie mriežky v grafe, či uloženie textových výstupov do formátov `txt`, či `JSON`. Zmena chronologického poradia výpisov vo výstupných oknách je umožnená interaktívnym objektom na zvolenie poradia od najnovšieho, resp. od najstaršieho záznamu.

Manažment interakcie užívateľa. Prepojenie ovládacích prvkov s výpočtovou časťou programu je zabezpečené sledovacími metódami združenými pod jeden kontajnerový objekt. Princíp týchto metód je uložiť preferenciu užívateľa a na jej základe jej hodnoty rozhodnúť o spustení patričných metód a funkcií iných častí programu. Príklad najčastejšieho scenára pri použití sledovacej metódy je vyžiadanie zobrazenia niektorej vlastnosti priebehu funkcie od užívateľa, zmena jej parametra `visible`, čo ovplyvní jej následné vykreslenie a následné



Obr. 3.1: Základný návrh grafického rozhrania

vypísanie sprievodného textu do výstupných okien.

3.3 Konfigurácia a spúšťanie programu

Niektoré výučbové situácie si vyžadujú upravenie základných parametrov programu, resp. potlačenie jeho prednastavenej funkcionality. Najmä grafické parametre samotného grafu, ako hrúbka a štýl krivky, či veľkosť vykresleného bodu, môžu od prípadu k prípadu vyžadovať ich úpravu. Podobne niektoré minimálne, či maximálne hodnoty intervalov, z ktorých si môže užívateľ vyberať pri vyšetrovaní priebehu funkcie.

Spúšťanie inštancie grafického programu v prostredí Jupyter Notebook si vyžaduje zvýšenú pozornosť. Okrem samotnej kontroly užívateľského vstupu je dôležité najmä správne referovať na vytvorené grafy, aby nedochádzalo ku kolíziám vypočítaných údajov pri viacerých inštanciách programu v jednom dokumente.

Požiadavky pre oblasť konfigurácie a spúšťania programu sa dajú zhrnúť nasledovne:

1. vytvorenie a umožnenie modifikovania konfiguračného súboru;
2. načítavanie a kontrola užívateľom definovaných údajov;
3. úvodný prepočet dát pre okamžitú funkčnosť programu;
4. spúšťanie programu v jednej bunke prostredia Jupyter Notebook;
5. manažment viacerých inštancií programu v jednom dokumente.

Konfiguračný súbor. Tak, ako celé grafické prostredie programu, aj pri výbere formátu konfiguračného súboru sú kladené požiadavky na jeho prehľadnosť, čitateľnosť a ľahkú modifikáciu. Uvažovaná je jeho jednoduchá konverzia do nejakej dátovej štruktúry jazyka Python z dôvodu plynulého použitia všetkými časťami programu.

Tento konfiguračný súbor je súčasťou priečinka so zdrojovými súbormi a je tak prístupný pre užívateľa, pričom sa predpokladá jeho validácia programom, t.j. vyhodnotenie prípadných nepovolených parametrov a vypísanie príslušného upozornenia.

Ďalej sa predpokladá globálna inštancia konfiguračného objektu tvoreného údajmi z konfiguračného súboru, tak, aby bola prístupná pre všetky časti programu.

Spolu so zdrojovými súbormi je konfiguračný súbor pripravený v prednastavenej forme a nevyžaduje dodatočné zásahy pred spustením programu.

Spúšťanie programu a kontrola vstupu. Spúšťanie programu v bunke prostredia Jupyter Notebook by malo byť čo najintuitívnejšie. Preto sa na jeho spustenie používa jediná funkcia `editor` s niekoľkými povinnými parametrami, ktorá po svojom spustení stále vytvorí novú inštanciu programu. Je tým zabezpečené použitie viacerých editorov v jednom dokumente prostredia Jupyter Notebook.

Vytváranie inštancie editora je podmienené kontrolou vstupných parametrov od užívateľa, t.j. či sú zadané v správnych formátoch, resp. či sú zadané všetky povinné parametre. Medzi povinné parametre patria objekty `matplotlib.Figure` a `matplotlib.Axes`, a to vzhľadom na čo najväčšiu možnosť dedenia parametrov vopred pripraveného grafu od užívateľa. Ďalšími povinnými parametrami sú predpis funkcie a zoznam intervalov, na ktorých sa bude táto funkcia vyšetrovať. Nepovinným parametrom sú definície (predpisy) derivácií. Všetky predpisy funkcií alebo derivácií musia byť v `ufunc` verzii, t.j. pracovať naraz s viacerými hodnotami vo forme vektorov. Program sa spustí v prípade poskytnutia všetkých povinných parametrov a ich správnosti a vyhlási chybu, ak tomu tak nie je.

Okamžitý výpočet dát. Pri vytváraní inštancie, t.j. inicializácii pracovnej plochy a iných potrebných štruktúr, sa počíta s okamžitým vykreslením grafu. Prvé prepočty údajov tak prebiehajú už pri vytváraní danej inštancie programu a užívateľovi je tak následne umožnená plná funkcionalita, bez potreby dodatočného nastavovania.

4 Implementácia

Vo výučbovom procese laboratórií predmetu *Matematická analýza I* študenti definujú funkcie a programujú grafy v jazyku Python v rámci výpočtového prostredia Jupyter. Použitie tohto výpočtového prostredia so sebou nesie isté obmedzenia v podobe vytvárania interaktívnych grafických prvkov. Aby mohli byť tieto prvky použité na vytvorenie grafického užívateľského rozhrania, je nevyhnutné použiť objekty knižnice ipywidgets. Druhým obmedzením vyplývajúcim z prostredia Jupyter je použitie knižnice matplotlib na vizualizáciu grafov funkcií. Obidve tieto knižnice sú totiž napísané práve v jazyku Python, preto je jeho použitie ako programovacieho jazyka pre implementáciu tejto práce nevyhnutné.

V tejto kapitole sú prezentované konkrétne implementačné techniky použité vo všetkých navrhnutých moduloch (kapitola 3). V časti 4.1 je popísaná implementácia hlavnej dátovej štruktúry reprezentujúcej užívateľskú funkciu, použitie numerických algoritmov pre výpočet derivácií a nulových bodov, či implementácia viet o vyšetrovaní priebehu funkcie pomocou diferenciálneho počtu. Spôsob naprogramovania grafického užívateľského rozhrania je prezentovaný v časti 4.2 a implementácia konfigurácie programu spolu s jeho spúšťaním (4.3) tvorí záver tejto kapitoly.

4.1 Modul pre analýzu funkcií

Užívateľská funkcia. ...

Výpočet derivácií a nulových bodov. ...

Výpočet vlastností priebehu funkcie. ...

4.2 Modul pre grafické užívateľské rozhranie

Pracovná plocha a výstupné okná. ...

Hlavné menu. ...

Interakcia s užívateľom. ...

4.3 Moduly pre konfiguráciu a spúšťanie programu

UNKNOWN. ...

UNKNOWN. ...

5 Testovanie a používateľská príručka

...

5.1 Sekcia 1

...

5.2 Sekcia 2

...

Záver

TODO

Literatúra

[1] *Anaconda Distribution Documentation*.
URL: `docs.anaconda.com/anaconda` (citované: 11.2.2020).

[2] ATKINSON, Kendall E. *An Introduction to Numerical Analysis*.
Wiley, New York, Second Edition, 1989.

[3] ČERNÝ, Ilja. *Úvod do inteligentního kalkulu (1000 příkladů z elementární analýzy)*.
Akademie věd České republiky, 2002.

[4] The MathWorks, Inc. *MATLAB® Primer, Twenty-third printing*.
URL: `mathworks.com/help/releases/R2014b/pdf_doc/matlab/getstart.pdf` (citované: 10.2.2020).

[5] ELIÁŠ, Jozef. *Matematika (Úvod do numerickej analýzy)*.
Slovenská vysoká škola technická v Bratislave, 1974.

[6] *Jupyter Widgets Developer Docs*.
URL: `ipywidgets.readthedocs.io` (citované: 11.2.2020).

[7] KLUVÁNEK, Igor, MIŠÍK, Ladislav, ŠVEC, Marko. *Matematika I pre štúdium technických vied*. Slovenské vydavateľstvo technickej literatúry, 1959.

[8] KUBÁČEK, Zbyněk, VALÁŠEK, Ján. *Cvičenia z matematickej analýzy I*.
Univerzita Komenského v Bratislave, 1989.

[9] LACHNIET, Jason. *Introduction to GNU Octave, Second Edition*.
lulu.com, Inc, 2019.

[10] LEVY, Doron. *Introduction to Numerical Analysis I - Lecture Notes*.
University of Maryland, 2016.

[11] McKINNEY, Wes. *Python for Data Analysis, Second Edition*.
O'Reilly Media, Inc, 2017.

[12] MOIN, Parviz. *Fundamentals of Engineering Numerical Analysis, Second Edition*.
Cambridge University Press , 2010.

[13] PÉREZ, Fernando, GRANGER, Brian E. *IPython: A System for Interactive Scientific Computing, Computing in Science and Engineering, vol. 9, no. 3, pp. 21-29, 2007.* URL: `ipython.org` (citované: 10.2.2020).

[14] ROSSANT, Cyrille. *IPython Cookbook, Second Edition*. 2018. URL: `ipython-books.github.io` (citované: 11.2.2020).

[15] *SciPy v1.4.1 Reference Guide*. URL: `docs.scipy.org/doc/scipy/reference` (citované: 11.2.2020).

[16] *SymPy, Python library for symbolic mathematics*. URL: `sympy.org`

[17] TOSI, Sandro. *Matplotlib for Python Developers*. Packt Publishing Ltd., 2009.

[18] *WolframAlpha, Computational Intelligence*. URL: `wolframalpha.com`