Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Interaktívne vyšetrovanie priebehu elementárnych funkcií

Bakalárska práca

2023
Martin Letenay

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

INTERAKTÍVNE VYŠETROVANIE PRIEBEHU ELEMENTÁRNYCH FUNKCIÍ

BAKALÁRSKA PRÁCA

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

Bratislava, 2023
Martin Letenay

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Martin Letenay
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. Tento nástroj má edukačný charakter a je určený študentom úvodného kurzu matematickej analýzy na našej fakulte. Ako implementačný jazyk je zvolený programovací jazyk Python v prostredí Jupyter Notebook. Výsledná implementácia je publikovaná v niektorej zo slobodných licencií kompatibilnej s GNU GPLv3+.

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

Literatúra: Cvičenia z matematickej analýzy I / Zbyněk Kubáček, Ján Valášek. Bratislava : Univerzita Komenského, 2009.

Matematická analýza I / Tibor Neubrunn, Jozef Vencko. Bratislava : Univerzita Komenského, 1992.

Data Structures and Algorithms in Python / Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser. Wiley, 2013.

Learning IPython for Interactive Computing and Data Visualization / Cyrille Rossant. Packt Publishing, 2nd edition, 2015.

Vedúci: Ing. Ján Komara, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 22.09.2022

Dátum schválenia: 26.09.2022 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Ďakujem môjmu školiteľovi Ing. Jánovi Komarovi, PhD za cenné rady, trpezlivosť a ochotu počas tvorby tejto práce.

Abstrakt

Cieľom tejto práce je implementovať interaktívny editor na vyšetrovanie priebehu elementárnych funkcií. Editor dokáže graficky vizualizovať vlastnosti elementárnych funkcií. Je primárne určený pre študentov úvodného kurzu Matematickej analýzy. Implementovaný je v jazyku Python a používaný v prostredí Jupyter Notebook.

Kľúčové slová: Matematická analýza, elementárna funkcia, vlastnosti, derivácia, Python, Jupyter Notebook

Abstract

The goal of this thesis is to implement interactive editor used for exploring properties of elementary functions. It can graphically visualise properties of elementary functions. It is primarily designed for students of the introductory course of Mathematical analysis. It is implemented in Python programing language and used in Jupyter Notebook computing environment.

Keywords: Mathematical analysis, elementary function, properties, derivative, Python, Jupyter Notebook

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Študenti Aplikovanej informatiky na našej fakulte musia v letnom semestri prvého ročníka absolvovať predmet Matematická analýza. Vrámci tohto predmetu sa stretnú s pojmami funkcia, limita, derivácia. Jednou z úloh, ktorú budú študenti riešiť, bude vyšetrovanie priebehu a určovanie základných vlastností funkcie. Väčšina vlastností, ktoré budú študenti vyšetrovať, sa dá určiť pomocou derivácie funkcie. Študenti sa s pojmom derivácie stretnú až v záverečnej časti semestra. Preto je nutné vytvoriť nástroj, ktorý im dovtedy s vyšetrovaním priebehu funkcie pomôže.

Cieľom tejto práce je navhnúť a implementovať interaktívny editor, ktorý dokáže vizualizovať graf funkcie a následne graficky zvýrazniť požadovanú vlastnosť. Študent tak dokáže lepšie pochopiť význam danej vlastnosti. Editor je implementovaný v jazyku Python a vložený do interaktívneho prostredia Jupyter Notebook. Študenti sa s jazykom Python stretnú už v zimnom semestri, a preto nebudú mať problém meniť parametre, ktoré editor pre svoje fungovanie potrebuje.

V prvej kapitole detailnejšie popíšeme ciele práce, určíme, ktoré vlastnosti funkcií budeme vyšetrovať. Vyšetrenie priebehu funkcie ukážeme na konkrétnom príklade, pomocou editora, aj matematicky.

V druhej kapitole predstavíme teoretické východiská práce. Zavedieme všetky pojmy a definície, ktoré budeme v práci používať, ako aj vety, podľa ktorých budeme určovať vlastnosti funkcie. Na základe týchto viet budeme implementovať výpočtové algoritmy.

V tretej kapitole popíšeme návrh celého systému, jeho funkcionalitu a navrhneme používateľské rozhranie. Predstavíme tiež potrebné knižnice a nástroje, ktoré využijeme pri implementácii editora.

V štvrtej kapitole sa venujeme implementácii. Predstavujeme jednotlivé triedy a metódy a popisujeme výpočtové algoritmy a spôsob ich vzájomnej interakcie.

1 Motivácia

V tejto kapitole vysvetlíme ciele tejto práce a dôvody jej vzniku. Cieľom tejto práce je vytvoriť softvér, ktorý sa bude využívať na hodinách predmetu Matematická analýza. Jedným z problémov, s ktorými sa v matematickej analýze stretávame, je vyšetrovanie priebehu funkcií. V tejto práci budeme pracovať iba s reálnymi funkciami s jednou premennou. Softvér, ktorý v rámci tejto práce vyvíjame, bude využívať algoritmy, ktoré vznikli použitím základných definícií a viet používaných v Matematickej analýze. Pri skúmaní priebehu funkcie môžeme pozorovať viaceré vlastnosti funkcie na zvolenom intervale. Vlastnosti, ktoré bude vyvíjaný softvér skúmať, sú **nulové body, ostré lokálne extrémy, intervaly monotónnosti, intervaly konvexnosti a konkávnosti a inflexné body**. Nebudeme skúmať inverzné funkcie ani asymptoty grafu funkcie. Vyšetrenie priebehu elementárnej funkcie budeme ilustrovať na konkrétnom príklade.

1.1 Nakreslenie grafu funkcie ma milimetrovom papieri

Máme elementárnu funkciu f danú predpisom

$$f(x) = \frac{x^2}{2} - 3x - \ln(\sqrt{x^2+1}) + 5\arctan(x)$$

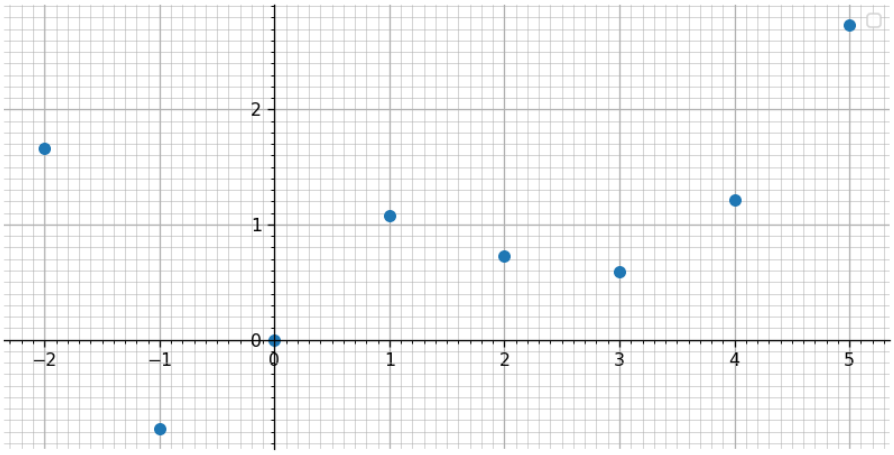
Táto elementárna funkcia obsahuje logaritmickú, mocninovú aj cyklometrickú funkciu a má niekoľko zaujímavých vlastností. Predtým, ako začneme vyšetrovať priebeh, je vhodné načrtnúť graf funkcie. Načrtneme ho napríklad pre hodnoty x z intervalu $\langle -2,5 \rangle$ na milimetrovom papieri, aby sme z grafu vedeli odhadnúť potrebné hodnoty. Predtým vytvoríme tabuľku bodov a ich odpovedajúcich funkčných hodnôt.

x	-2	-1	0	1	2	3	4	5
f(x)	1.660	-0.774	0.000	1.080	0.731	0.594	1.212	2.738

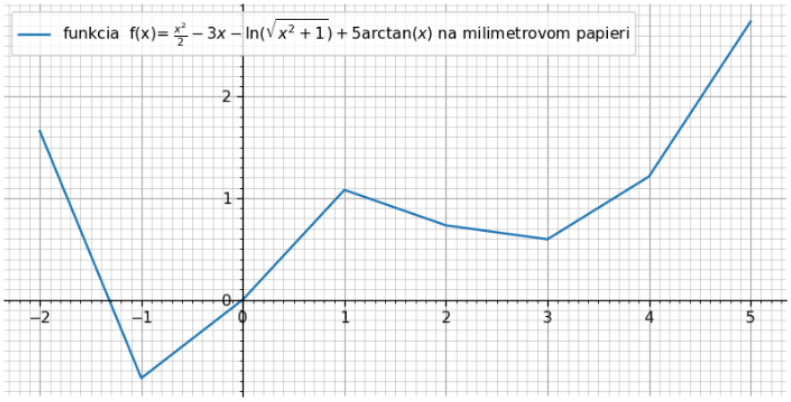
Tabuľka 1: Tabuľka hodnôt

Tieto body môžeme pospájať úsečkami, aby sme mali predstavu o tom, ako sa naša funkcia na danom intervale správa.

Takýto graf slúži len na ilustráciu. Aby sme mohli vyšetrovať priebeh, musíme ho načrtnúť presnejšie. Využijeme pritom pythonovské knižnice Numpy a Matplotlib v prostredí



Obr. 1: Graf funkcie $f(x) = \frac{x^2}{2} - 3x - \ln(\sqrt{x^2+1}) + 5\arctan(x)$ na milimetrovom papieri



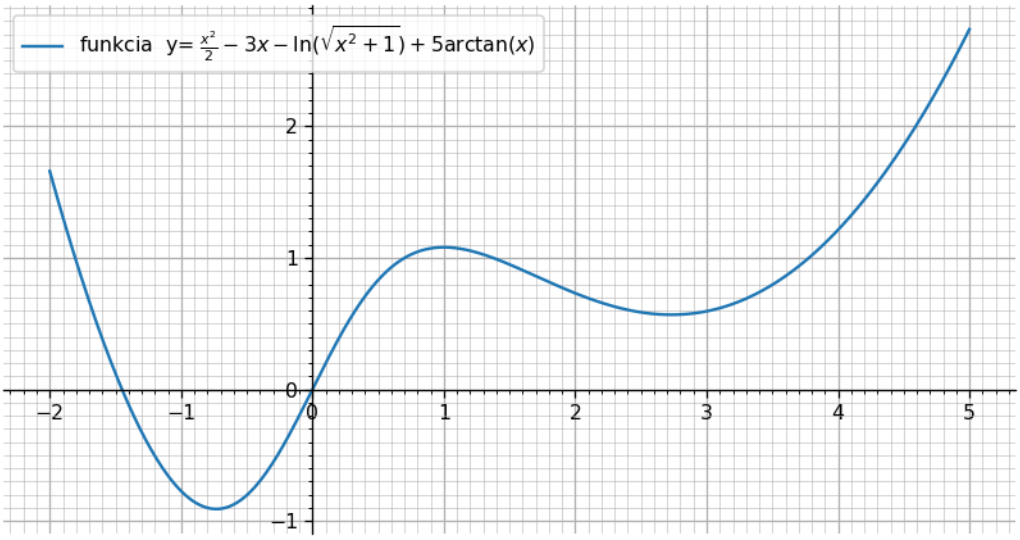
Obr. 2: Graf funkcie vytvorený pospájaním bodov

Jupyter Notebook. Pomocou Numpy vieme vytvoriť interval napríklad s 500 bodmi a určiť mu aj krajné body. Graf nakreslíme pomocou kódu na nasledujúcom obrázku.

Tento kód vygeneruje nasledovný graf. Z takéhoto grafu vieme, že funkcia má niekoľko nulových bodov aj extrémov na danom intervale. Pri vyšetrovaní priebehu sa sústredíme na ostré lokálne extrémy a monotónnosť. Jedno ostré lokálne maximum sa nachádza v bode 1. Ostré lokálne minimá sú dve. Jedno na intervale $\langle -1, 0 \rangle$ a druhé na intervale $\langle 2, 3 \rangle$. Na zistenie približných súradníc použijeme editor.

```
#### vstupné údaje

## funkcia
def f(X): return X ** 2 / 2 - 3 * X - np.log(np.sqrt(X ** 2 + 1)) + 5 * np.arctan(X)
X = np.linspace(-2, 5, 5*100+1)

#### obrázok s dvoma diagramami
fig, ax = plt.subplots()
fig.set_size_inches(9, 5)

### spoločné nastavenie pre osi
ax.set_aspect('equal')

### 1. diagram
init_subplot(ax)

ax.xaxis.set_major_locator(MultipleLocator(1))
ax.yaxis.set_major_locator(MultipleLocator(1))
ax.grid(which='major')
ax.xaxis.set_minor_locator(MultipleLocator(1/10))
ax.yaxis.set_minor_locator(MultipleLocator(1/10))
ax.grid(which='minor', linewidth=1/3)
## graf funkcie
ax.plot(X, f(X), label=r"funkcia  y= $\frac{{x^2}}{2} - 3x - \ln(\sqrt{{x^2 + 1}}) + 5 \arctan(x)$")
ax.legend()
fig.show()
```

Obr. 3: Kód na nakreslenie grafu



Obr. 4: Graf funkcie $f(x) = \frac{x^2}{2} - 3x - \ln(\sqrt{x^2+1}) + 5\arctan(x)$ na milimetrovom papieri

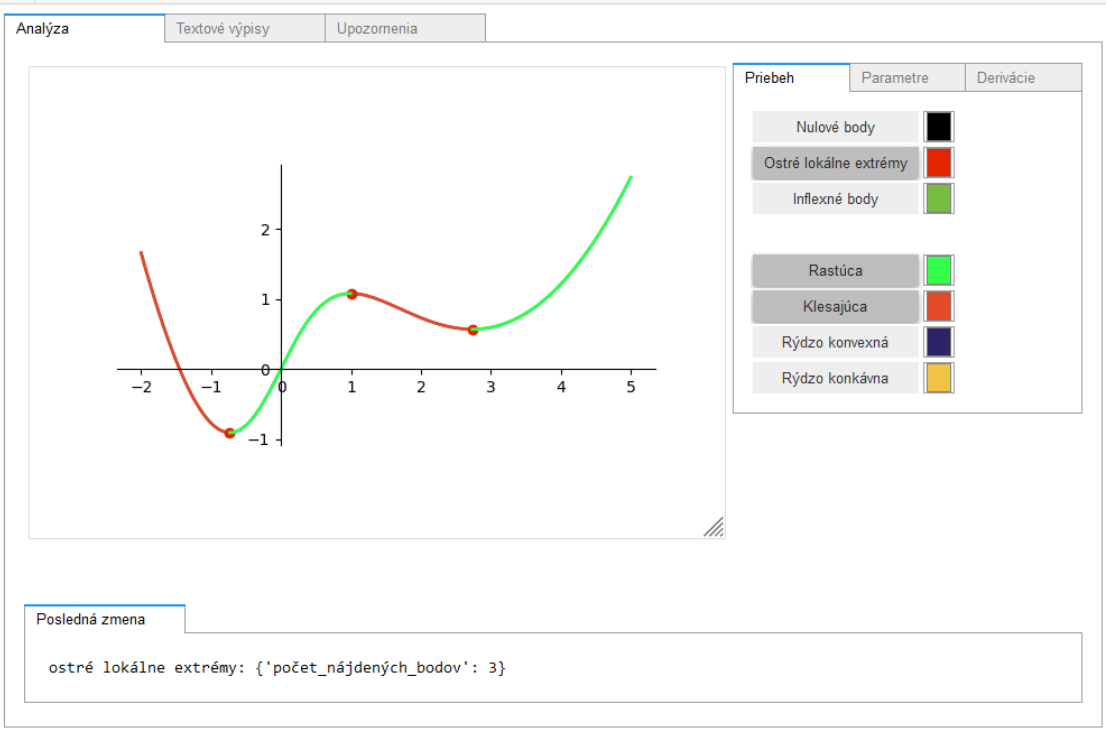
1.2 Vyšetrovanie priebehu funkcie pomocou editora

Teraz na vyšetrenie priebehu funkcie využijeme editor. Na jeho inicializáciu použijeme rovnaké parametre ako na nakreslenie grafu. Môžeme použiť aj predpis pre prvú deriváciu.

```
In [ ]:
## prvá derivácia
def df(X): return (X ** 3 -3 * X **2 +2) / (X ** 2 + 1)
dX = X

#### editor
run_editor(figure=fig, axes=ax, function=f, intervals=[X], primes=[df])
```

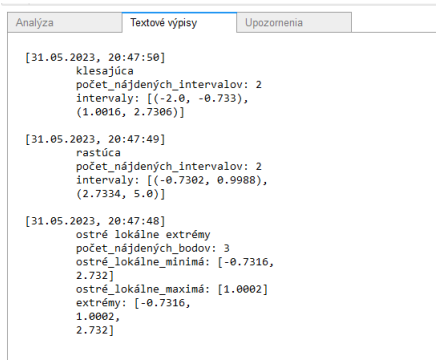
Obr. 5: Spustenie editora

Keď sa editor spustí, stlačíme tlačidlo Ostré lokálne extrémy. Editor ich vykreslí a zobrazí tiež informáciu, koľko extrémov našiel. Môžeme zobraziť aj intervaly monotónnosti. Ak chceme zistiť približné súradnice extrémov, klikneme na záložku Textové výpisy. Grafický výstup z editora je na obrázku.



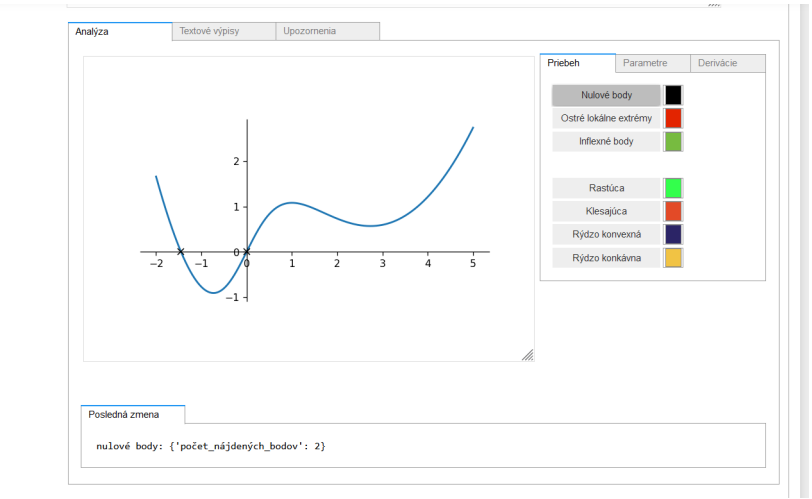
Obr. 6: Zobrazenie extrémov v editore aj s intervalmi monotónnosti

Hodnoty, ktoré vypočítal editor, sú len približné. Editor totiž pracuje s určitou presnosťou, extrémy hľadá pomocou derivácií, pričom nulové body derivácií aproximuje. Študent sa však na hodinách stretne s editorom skôr, než sa oboznámi s vyšetrovaním priebehu funkcie pomocou derivácie. Preto sú aj približné hodnoty postačujúce. Výpočet extrémov pomocou derivácie ukážeme v nasledujúcej časti.

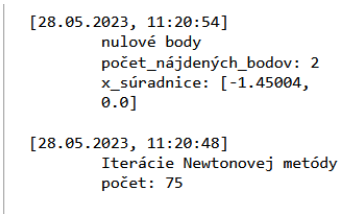


Obr. 7: Zobrazenie extrémov v editore (Textové výpisy)

V editore sa vrátime do záložky Analýza a necháme zobraziť nulové body. Súradnice zistíme znova z výpisov.



Obr. 8: Zobrazenie nulových bodov v editore



Obr. 9: Nulové body

Okrem priebehu funkcie môžeme v editore zobrazovať derivácie funkcií, alebo nastavovať parametre, koré ovplyvňujú výpočty hodnôt. Presný popis metód použitých na vyšetrovanie priebehu funkcie ako aj ich implementácia sú popísané v nasledujúcich kapitolách.

1.3 Vyšetrovanie priebehu funkcie pomocou derivácie

Z grafu vidíme, že funkcia má maximum v bode 1. Okrem neho má ešte dve minimá. Ich súradnice vypočítame pomocou nulových bodov prvej derivácie našej funkcie. Derivácia má predpis:

$$y' = \frac{x^3 - 3x^2 + 2}{x^2 + 1}$$

Súradnice extrémov sú riešenia rovnice

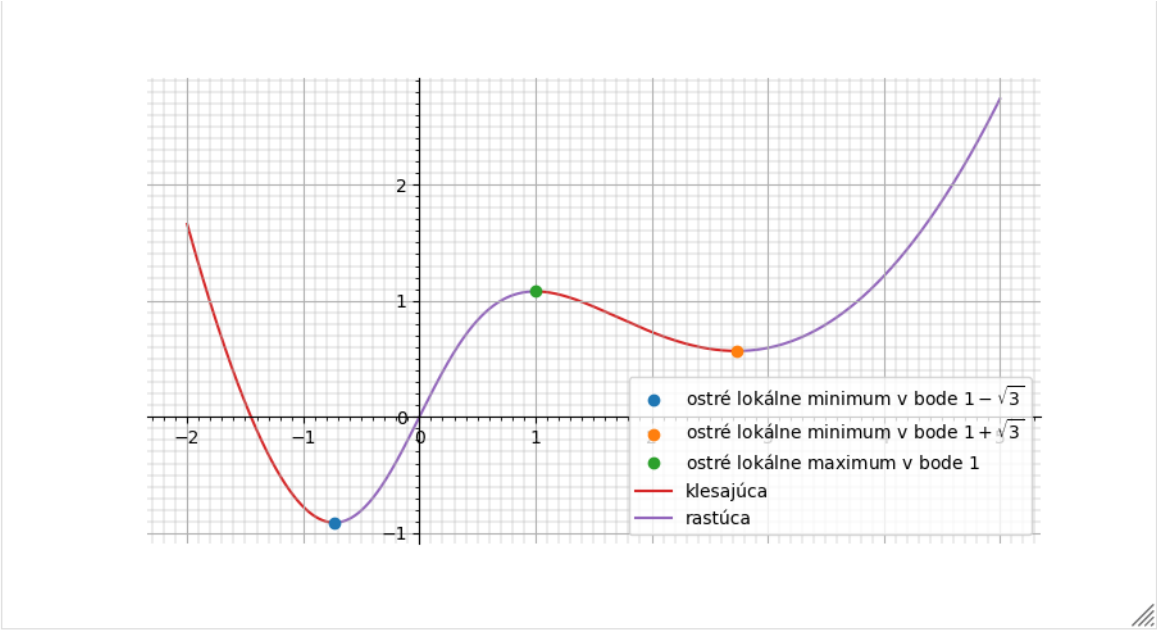
$$x^3 - 3x^2 + 2 = 0$$

túto rovnicu rozpíšeme ako

$$(x-1)(x^2 - 2x - 2) = 0$$

a získame riešenia $x_1 = 1, x_2 = 1 - \sqrt{3}, x_3 + \sqrt{3}$

Pomocou extrémov vieme tiež určiť, kde funkcia rastie alebo klesá (intervaly monotónnosti). Ak chceme tieto výsledky znázorniť graficky, dostaneme nasledujúci obrázok.



Obr. 10: Priebeh funkcie $f(x) = \frac{x^2}{2} - 3x - \ln(\sqrt{x^2+1}) + 5\arctan(x)$

2 Východiská práce

Táto kapitola vysvetľuje základné pojmy požívané v tejto práci. Taktiež krátko popisuje iné dostupné systémy a programy s podobnou funkcionalitou.

2.1 Matematická analýza

V tejto časti popíšeme definície pojmov Matematickej analýzy.Vychádzame z použitej literatúry. [3, 4]

2.1.1 Funkcia, definičný obor a obor hodnôt

Definícia Nech A, B sú dve množiny. Ak ku každému prvku $x \in A$ priradíme práve jeden prvok $y \in B$ hovoríme, že na množine A je definovaná funkcia nadobúdajúca hodnoty v množine B. Píšeme $y = f(x)$ a hovoríme o funkcii f (niekedy tiež o zobrazení f). Zobrazenie definované na A s hodnotami v B označujeme tiež takto: $f : A \to B$. Množinu A nazývame definičným oborom funkcie f. Označujeme ho $D(f)$. Množinu, pozostávajúcu zo všetkých tých prvkov $y \in B$, ktoré sú priradené prvkom $x \in A$, nazývame oborom hodnôt funkcie f. Prvky z definičného oboru nazývame často vzormi, prvky z oboru hodnôt obrazmi. Niekedy sa premenným $x \in A$ hovorí nezávislé premenné a k nim priradeným premenným $y \in B$ závislé premenné.

2.1.2 Elementárne funkcie

Definícia Obyčajne sa termínom „elementárne funkcie“ označujú: všeobecná mocnina, exponenciálne funkcie, logaritmické funkcie, goniometrické funkcie a cyklometrické funkcie, ako aj funkcie, ktoré sa dajú získať z týchto funkcií pomocou štyroch základných algebraických operácií a tvorením zložených funkcií.

2.1.3 Derivácia funkcie

Definícia Deriváciou funkcie f rozumieme funkciu f', ktorej hodnota $f'(x)$ vo vnútornom bode x definičného oboru funkcie f je smernicou dotyčnice grafu funkcie f v bode $[x, f(x)]$.

Formálne

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

2.1.4 Nulové body

Definícia Hodnoty x z definičného oboru funkcie f, kde $f(x) = 0$ sa nazývajú nulové body funkcie. Určujú, kde sa nachádzajú priesečníky grafu funkcie f s osou x.

2.1.5 Ostré lokálne extrémy

Definícia Hovoríme, že funkcia f má v bode x_0 ostré lokálne maximum (ostré lokálne minimum), ak existuje okolie $O(x_0)$ bodu x_0 také, že pre všetky $x \in O(x_0) \cap D(f), x \neq x_0$ platí

$$f(x) < f(x_0) \qquad (f(x) > f(x_0)).$$

Ostré lokálne maximá a minimá sa spoločným termínom označujú ako ostré lokálne extrémy.

Veta *Nech funkcia f vyhovuje vo vnútornom bode $x_0 \in D(f)$ podmienkam:*

1. $f'(x_0) = f''(x_0) = \cdots = f^{(n-1)}(x_0) = 0$,
2. $f^{(n)}(x_0) \neq 0$.

Potom pri párnom n má funkcia f v bode x_0 ostrý lokálny extrém, teda buď minimum, ak $f^{(n)}(x_0) > 0$, alebo maximum, ak $f^{(n)}(x_0) < 0$. Pri nepárnom n funkcia f nemá v bode x_0 ostrý lokálny extrém.

2.1.6 Intervaly monotónnosti

Definícia Nech $A \subset D(f)$. Funkciu f budeme nazývať rastúcou (klesajúcou) na množine A, ak pre každé $x_1, x_2 \in A$ také, že $x_1 < x_2$, platí $f(x_1) < f(x_2)$ $(f(x_1) > f(x_2))$. Funkcie rastúce a klesajúce nazývame rýdzomonotónnymi.

Veta *Nech funkcia f je spojitá na intervale I a má deriváciu v každom jeho vnútornom bode Potom ak prvá derivácia je kladná, tak f je rastúca na I. Ak je záporná, tak f je klesajúca na I.*

2.1.7 Intervaly rýdzej konkávnosti a konvexnosti

Definícia Budeme hovoriť, že funkcia f je rýdzo konvexná na intervale I ak pre všetky $x_1, x_2 \in I$, a všetky $p, q > 0, p + q = 1$ platí

$$f(px_1 + qx_2) < pf(x_1) + qf(x_2)$$

Hovoríme, že funkcia f je rýdzo konkávna na intervale I, ak platí obrátená nerovnosť k prechádzajúcej definícii.

Veta *Nech funkcia f je spojitá na intervale I a dvakrát diferencovateľná v každom jeho vnútornom bode. Potom ak druhá derivácia je kladná, tak f je rýdzo konvexná na I. Ak je záporná, tak f je rýdzo konkávna na I.*

2.1.8 Inflexné body

Definícia Vnútorný bod x definičného oboru funkcie sa nazýva inflexný bod funkcie, ak funkcia má v bode x deriváciu a existuje $h > 0$ tak, že funkcia je rýdzo konvexná na jednej z množín $(x-h,x\rangle$, $\langle x,x+h)$ a rýdzo konkávna na druhej z nich.

Veta *Nech funkcia f je trikrát diferencovateľná v bode x a dvakrát diferencovateľná v niektorom jeho okolí. Ak platí:*

1. $f''(x) = 0$
2. $f'''(x) \neq 0$

potom funkcia f má v bode x inflexmý bod.

2.2 Numerická analýza

V tejto časti popíšeme metódy a pojmy numerickej analýzy, ktoré budeme používať. Vychádzame z použitej literatúry. [1, 6]

2.2.1 Tolerancia chyby a zaokrúhľovanie

Vypočítané hodnoty budeme zaokrúhľovať na určitý počet platných cifier, ktorý bude preddefinovaný. Numerické metódy, ktoré budeme používať na vyšetrovanie priebehu funkcie, budú pracovať s toleranciou chyby definovanou podľa nasledujúcej vety.

Veta Nech približná hodnota a čísla A má n platných cifier, Potom pre jej relatívnu chybu εa platí

$$\varepsilon a \leq \frac{1}{\alpha_m}(\frac{1}{10})^{n-1}$$

kde α_m je prvá platná cifra čísla a.

2.2.2 Aproximácia nulových bodov funkcie (Newtonova metóda)

Vo všeobecnosti je veľmi ťažké presne určiť nulové body funkcie. Dokážeme ich ale aproximovať. Z grafu odhadneme, že sa nulový bod náchádza v bode x_0. Zostrojíme dotyčnicu k funkcii f v bode x_0. Ak $f'(x_0) \neq 0$ tak dotyčinca pretína os x v bode x_1, ktorý je ku skutočnému nulovému bodu bližšie, než odhad v bode x_0 Tento proces zopakujeme a nájdeme bod x_2, takto pokračujeme a naše odhady $x_0, x_1, x_2 \cdots$ sa približujú ku skutočnému nulovému bodu x_n. Predpis dotyčnice k funkcii f v bode x_0 je daný rovnicou:

$$f(x_0) + f'(x_0)(x_1 - x_0) = 0$$

Z tejto rovnice vyjadríme x_1.

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

alebo vo všeobecnosti x_n pre $n > 0$

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

2.2.3 Numerické derivovanie

Deriváciu funkcie v bode $x \in D(f)$ budeme aproximovať metódou centrálnej diferencie.

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

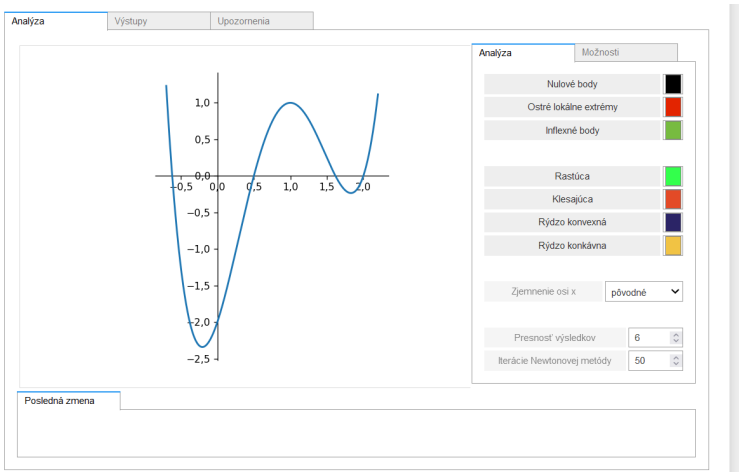
kde $h > 0$. Číslo h nazývame krokom x a predstavuje rozdiel dvoch susedných hodnôt z definičného oboru. Chyba výpočtu sa pri centrálnej diferencii zmenšuje vzhľadom na h^2.

2.3 Prehľad podobných systémov

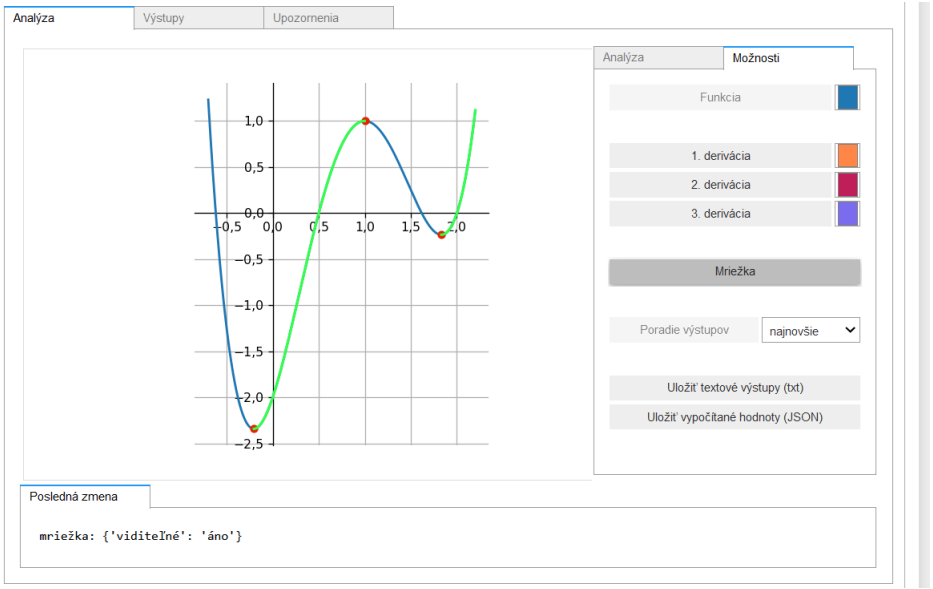
2.3.1 JEDIT

JEDIT je knižnica vytvorená bývalým študentom FMFI Jurajom Vetrákom. Knižnica je integrovaná priamo do prostredia Jupyter Notebook, ktoré sa používa na hodinách Matematickej analýzy. Po spustení sa zobrazí súradnicová sútava a základný graf funkcie. Má jednoduché používateľské rozhranie a ovláda sa pomocou tlačidiel, ktoré spúšťajú zobrazovanie jednotlivých vlastností funkcie. Výhodou je možnosť upraviť farbu zobrazenia vlastností podľa potrieb používateľa.

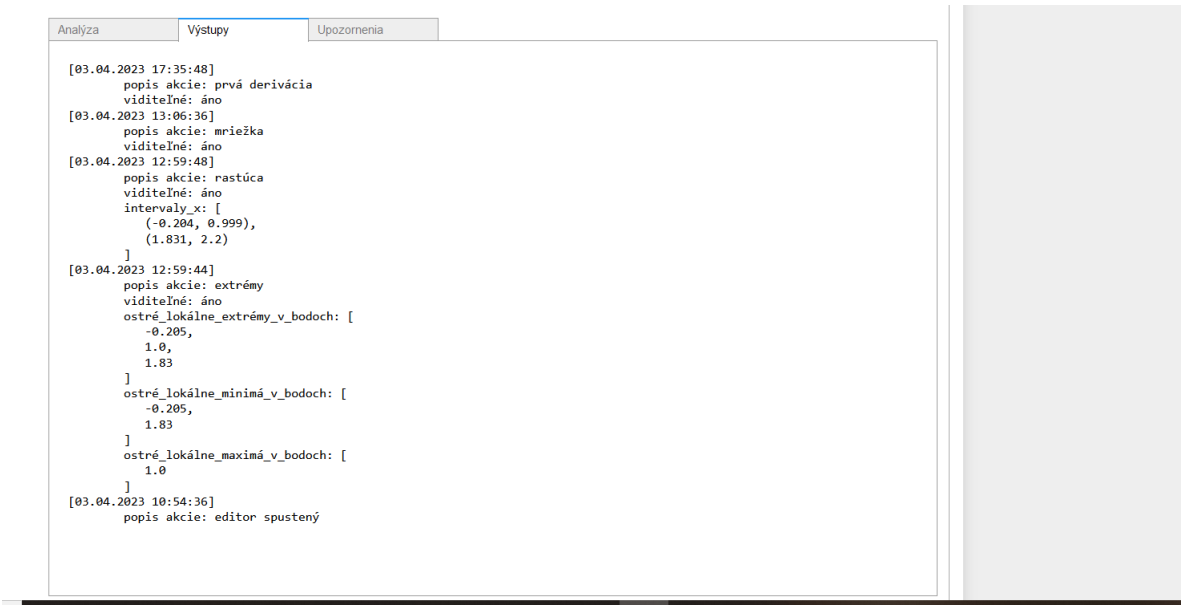
Ak používateľ zvolí zobrazenie vlastnosti, celý graf sa prekreslí a do malej záložky pod grafom sa vypíše popis poslednej vykonanej aktivity. Tiež je možné meniť jemnosť osi x alebo zobrazovať derivácie funkcie alebo zobraziť ku grafu mriežku pre lepšie odčítanie bodov. JEDIT tiež poskytuje možnosť textového zobrazenia vypočítaných hodnôt pre potreby ďalšieho využitia, uloženie hodnôt do súboru. Avšak pri ukladaní do súboru sa vyskytujú chyby.



Obr. 11: JEDIT - Základný graf



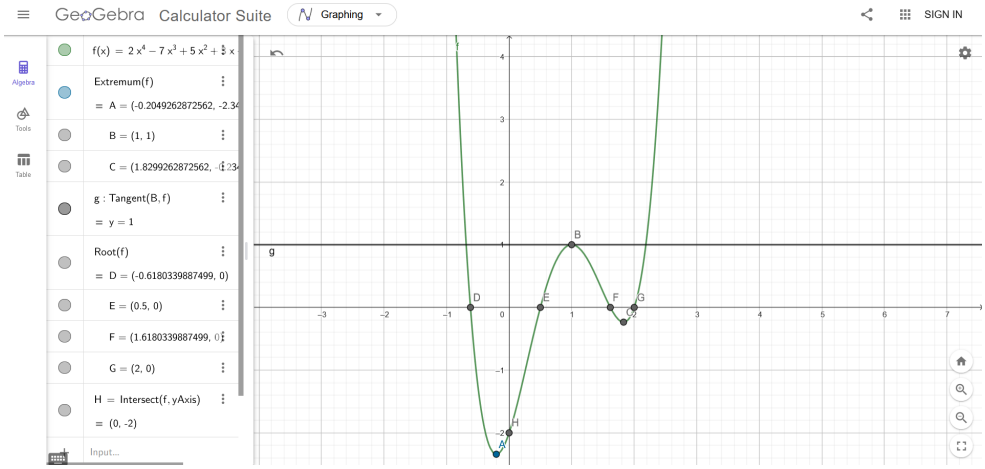
Obr. 12: JEDIT -Analýza vlastností funkcie z grafu



Obr. 13: JEDIT - Textové výstupy z editora

2.3.2 Geogebra graphing calculator

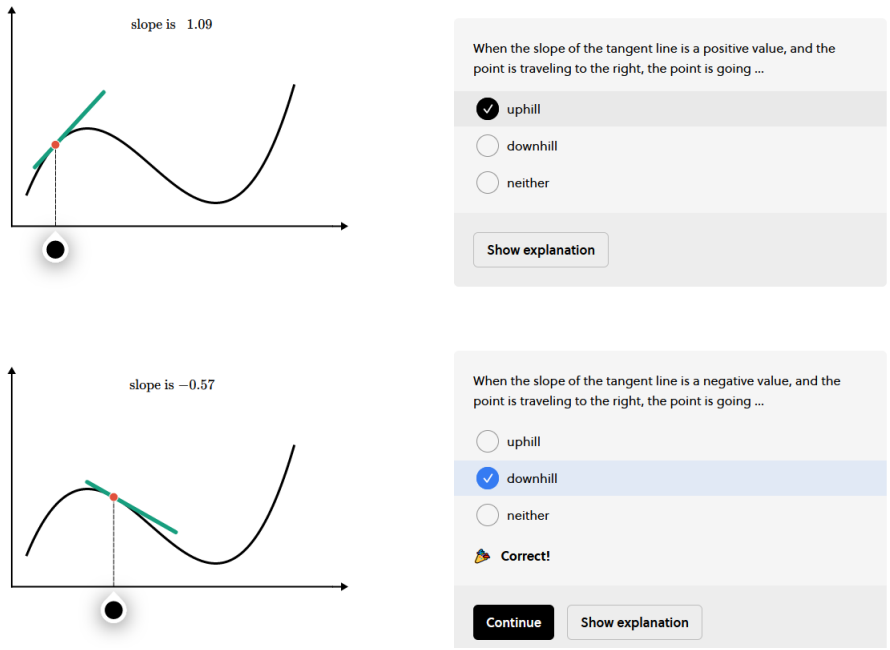
Geogebra je jeden z mnohých online nástrojov na kreslenie grafov. Neponúka však všetko, čo ponúka JEDIT. Vie zobraziť ostré lokálne extrémy a nulové body a zostrojiť dotyčnicu v nejakom bode. Nedajú sa ale nastaviť intervaly a možnosti prispôsobovania grafu sú skryté medzi nastaveniami a vyžadujú množstvo klikania. Preto je tento softvér vhodnejší na kreslenie grafov funkcíi než na vyšetrovanie ich priebehu. [13]



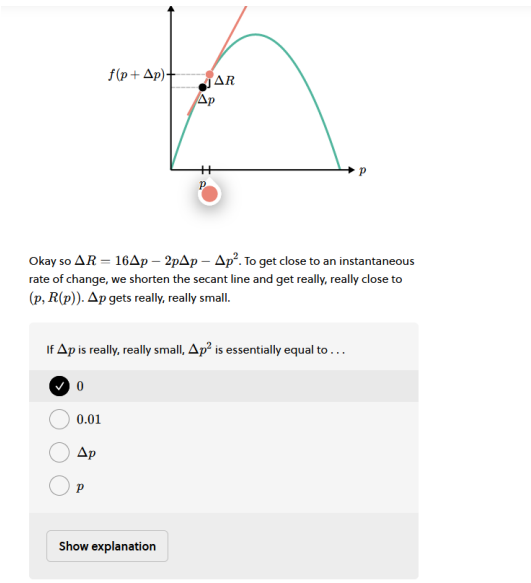
Obr. 14: Geogebra - ukážka

2.3.3 Brilliant

Brilliant je webový vzdelávací portál, ktorý umožňuje učiť sa interaktívnou formou. Ponúka veľké množstvo kurzov matematiky, fyziky a informatiky. Napríklad kurz Calculus in a nutshell, ktorý študentom vysvetlí základné princípy a pojmy, napríklad derivácie. alebo lokálne extrémy. Učivo je vysvetlené stručne, s pomocou jednoduchých slovných úloh alebo interaktívnych grafov. Veľkou nevýhodou je obmedzené množstvo lekcií v bezplatnom režime. [14]



Obr. 15: Brilliant - ilustrácia priebehu pomocou smernice dotyčnice



Obr. 16: Brilliant - ilustrácia vlastností derivácie

3 Návrh systému

V tejto kapitole postupne popíšeme, ako má byť výsledný softér navrhnutý, aby spĺňal všetky požiadavky.

3.1 Použité technológie

V krátkosti popíšeme technológie použité v tejto práci. [7, 8, 9, 10, 11, 12]

3.1.1 Jupyter notebook

Jupyter notebook je open-source webové prostredie na vizualizáciu dát a vedeckých výpočtov. Používateľ môže vytvárať dokumenty (notebooks), do ktorých môže vkladať časti kódu napísané v jazykoch Julia, Python alebo R a spúšťať ich alebo upravovať priamo z webového prehliadača. Taktiež vie zobrazovať čistý text alebo text vo formáte HTML, LaTeX alebo zobrazovať obrázky vo vysokej kvalite. Výhodou je ľahké upravovanie a zdieľanie kódu, dát a analýzy v jednom dokumente. Preto sa využíva pre edukačné aj výskumné účely. Jupyter notebooks môžeme po dokončení exportovať do statických formátov ako PDF, HTML alebo LaTeX.

3.1.2 Ipywidgets (Jupyter widgets)

Ipywidgets je knižnica pre jazyk Python, ktorá umožňuje pridávať interaktívne ovládacie prvky do prostredia Jupyter Notebook. Dostupné sú rôzne prvky ako textové polia, tlačidlá, zaškrtávacie polia a ďalšie, podobne ako pri HTML formulároch. Vďaka tejto knižnici sa prostredníctvom Jupyter Notevoook dajú vytvárať komplexné ale intuitívne aplikácie a dynamicky pracovať s vizualizovanými dátami. S jej pomocou budeme implementovať celé používateľské rozhranie.

3.1.3 Matplotlib

Matplotlib je knižnica pre jazyk Python, ktorá umožňuje vykreslovať grafy matematických funkcií alebo vizualizovať iné vedecké dáta. Obrázky vytvorené knižnicou Matplotlib možno

jednoducho vložiť do Jupyter notebookov bez akéhokoľvek zníženia kvality, alebo ich použiť do odborných publikácií. Matplotlib poskytuje množstvo možností ako vizualizovať grafy a meniť ich štýl a rozloženie pomocou interaktívnych prvkov.

3.1.4 Numpy

Numpy je knižnica pre jazyk Python na prácu s viacrozmernými homogénnymi poľami. Poskytuje rôzne vektorové aj skalárne operácie ako aj maskovanie polí. Aj keď je knižnica napísaná v Pythone, jej základ tvorí vysoko optimalizovaný kód v jazyku C. Vďaka tomu sú všetky funkcie rýchle a efektívne. Túto knižnicu budeme používať hlavne na rozdeľovanie vypočítaných hodnôt pre hľadanie význačných bodov použitím indexovania alebo matematických funkcií.

3.1.5 Scipy

Knižnica Scipy rozširuje možnosti knižnice Numpy a ponúka efektívne algoritmy na riešenie integrácie, interpolácie, diferenciálnych rovníc a iných problémov. Budeme z nej využívať napríklad implementáciu Newtonovej metódy a iné pomocné funkcie.

3.1.6 Findiff

Knižnica findiff poskytuje metódy na počítanie derivácií pre polia akýchkoľvek rozmerov, akéhokoľvek rádu. Používa vektorové operácie, čím zabezpečuje rýchlosť a efektivitu. Na výpočet hodnôt derivacií používa metódu centrálnej diferencie, ktorá bola popísaná v časti 2.2.

3.2 Vlastnosti vyvíjaného softvéru

V tejto časti stručne popíšeme vlastnosti a funkcie, ktoré by mal softvér vyvíjaný v rámci tejto práce mať.

3.2.1 Vizualizácia grafu funkcie a jej vlastností

Keďže editor bude integrovaný priamo do prostredia Jupyter Notebook, používateľ môže inicializovať editor pomocou jednoduchého príkazu, ktorý môže byť súčasťou väčšieho bloku kódu. Ak používateľ zadá parametre na inicializáciu nesprávne, objaví sa chybové hlásenie. Po úspešnom spustení sa vykreslí pravouhlá súradnicová sústava a graf funkcie. Následne má používateľ možnosť na danom obrázku vizualizovať priebeh funkcie.

3.2.2 Textové výstupy

Keď používateľ zvolí zobrazenie nejakej vlastnosti, táto vlastnosť sa zvýrazní na grafe a vypíše sa textový popis vlastnosti a vypočítané hodnoty, ktoré má používateľ k dispozícii pre ďalšie použitie. Formát textových výpisov popíšeme v návhru používateľského rozhrania.

3.2.3 Nastavovanie parametrov pre výpočty

Zjemnenie osi x Používateľ musí pri spustení zadať intervaly, pre ktoré chce funkciu zobraziť. Každý interval bude mať začiatočný a koncový bod a počet bodov daného intervalu. Ak bude chcieť používateľ získať presnejšie výsledky, bude môcť zmeniť počet bodov v každom intervale. Ak ale používateľ tento parameter zmení, všetky doteraz vypočítané hodnoty sa musia prepočítať.

Zjemnenie osi y Numerické metódy, ktoré bude softvér používať, operujú s vopred určenou presnosťou. Chceme docieliť, aby bol vyvíjaný softvér čo najuniverzálnejší. Niektoré funkcie môžu mať taký priebeh, že sa približujú lokálnemu extrému alebo nulovému bodu, ale nedosiahnu ho. Ak máme zle nastavenú presnosť, môže sa stať, že softvér zobrazí nejaký význačný bod na grafe, aj keď sa tam v skutočnosti nenachádza. S možnosťou meniť presnosť môže užívateľ získať presnejšie výsledky. Zmena tohto parametra spustí prepočet funkcie.

Zaokrúhlenie Vypočítané výsledky sa vždy zaokrúhlia na predvolený počet platných cifier. Používateľ môže túto hodnotu zmeniť a výsledky sa prepočítajú. Chyba výpočtu sa mení podľa vety v časti 2.2.1.

Iterácie Newtonovej metódy Používateľ bude môcť meniť počet iterácií Newtonovej metódy pre hľadanie nulových bodov popísanej v časti 2.2.2. Editor následne prepočíta približné súradnice a zobrazí nové výsledky.

Počet derivácií pre extrémy Používateľ bude môcť meniť počty derivácií pre hľadanie nulových bodov, ako bolo popísané v časti 2.1.5.

Presnosť numerického výpočtu derivácií Používateľ bude môcť meniť presnosť numerického výpočtu derivácií. Na výber bude mať 3 možnosti. ”Pôvodná”, ”presná” a ”veľmi presná”. Prepočítajú sa iba tie derivácie, ktoré neboli zadané používateľom.

3.3 Inicializačné parametre

Softvér bude na spustenie očakávať zadanie nasledovných parametrov od používateľa:

- figure - Objekt typu matplotlib.Figure, ktorý reprezentuje vykreslený obrázok.
- axes - Objekt typu matplotlib.Axes, ktorý reprezentuje súradnicové osi.
- function - pythonovská funkcia bez parametrov.
- intervals - zoznam (*list*) intervalov, kde sa má funkcia zobraziť.

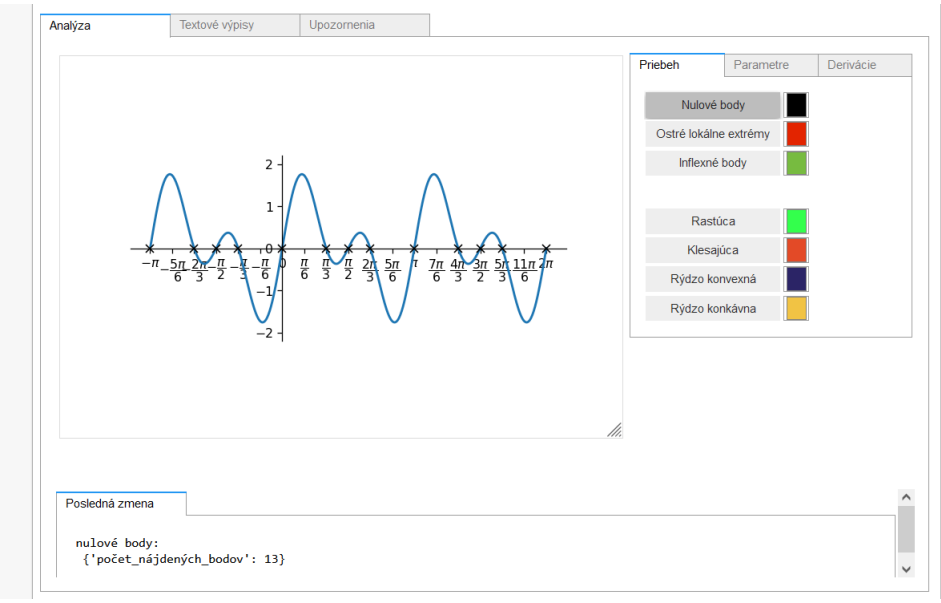
Taktiež je možné zadať pole derivácií funkcie ako nepovinný parameter. Má formát zoznam (*list*) predpisov derivácií rovnako ako pri parametri *function* v správnom poradí (prvá, druhá, tretia). Ak žiadne derivácie neboli zadané, softvér ich po spustení vypočíta numericky.

3.4 Používateľské rozhranie

V tejto časti stručne popíšeme návrh používateľského rozhrania. Používateľské rozhranie je rozdelené do troch častí.

- Analýza - hlavné okno editora, ktoré obsahuje obrázok grafu a používateľské rozhranie
- Textové výpisy - textové okno, kam sa vypisujú výsledky výpočtov (súradnice význačných bodov, intervaly alebo zvolené parametre)
- Upozornenia - textové okno, kam sa vypisujú prípadné upozornenia a chyby, vzniknuté počas behu programu, aby nevyrušovali používateľa pri práci s editorom.

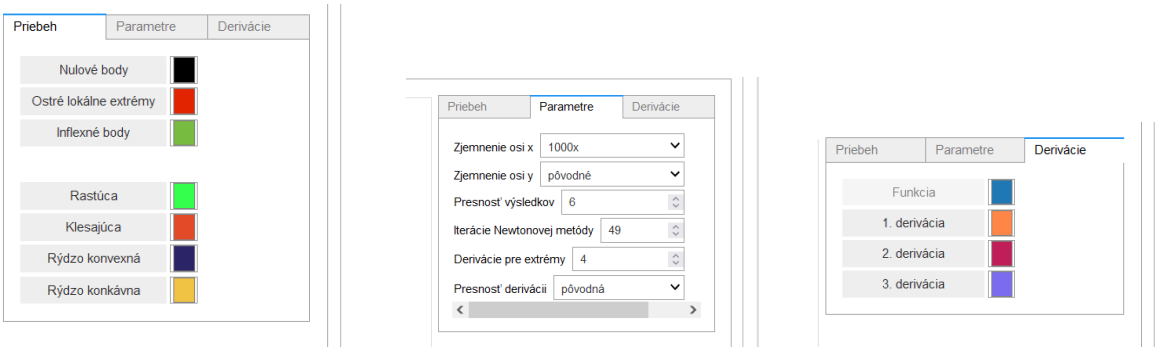
3.4.1 Hlavné okno



Obr. 17: Hlavné okno editora - funkcia $\sin(2x)+\cos(4x)$ a jej nulové body

Hlavné okno je rozdelené do troch častí. V strede je vykreslený samotný graf, vytvorený ako kópia grafu, ktorý vznikol na základe parametrov od používateľa. Všetky parametre, až na veľkosť obrázka, sú zadané používateľom. Pod obrázkom je malé textové okno slúžiace na popis poslednej vykonanej akcie, napríklad počet význačných bodov, ktoré editor našiel. Súradnice bodov sa vypíšu do okna Textové výpisy. Pri zvolení intervalu monotónnosti sa do textových výpisov vypíše otvorený interval obsahujúci iba krajné body. Vpravo vedľa grafu sa nachádajú ovládacie prvky.

3.4.2 Ovládacie prvky



(a) Ovládanie zobrazovania priebehu funkcie

(b) ovládanie parametrov pre výpočty

(c) ovládanie zobrazovania derivácii

Obr. 18: Ukážka ovládacích prvkov

Ovládanie editora je rozdelené intuitívne do troch záložiek.

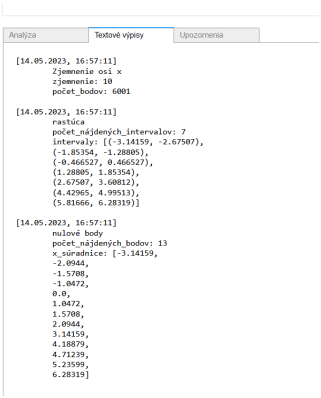
Prvá záložka s názvom Priebeh ovláda zobrazovanie význačných bodov a intervalov s možnosťou prispôsobiť farbu zobrazenia jednotlivých vlastností.

Druhá záložka s názvom Parametre ovláda funkcie, ktoré ovplyvňujú výpočty. Parametre ako zjemnenie osí sú riešené pomocou prvkov typu Dropdown, ktoré ponúkajú výber zo zoznamu možností a máme na výber hodnoty mocniny čísla 10. Pri zjemnení osi x vieme zjemnenie iba zvyšovať, pričom zjemnenie osi y vieme aj znižovať. Rovnako vieme meniť presnosť výpočtu derivácií. Keďže tieto parametre menia globálne parametre, musia sa prepočítať všetky hodnoty. Editor musí byť navrhnutý tak, aby sa prepočty udiali len na pozadí, aby to nevyrušovalo používateľa. Ten dostane iba správu o tom, že prebiehajú výpočty. Po skončení výpočtu sa vymaže pôvodný obsah okna Textové výpisy a nové hodnoty sa vypíšu len pre tie vlastnosti, ktoré má používateľ práve zvolené. Ostatné parametre, ktoré ovládajú výpočty iba jednej vlastnosti (nulové body, ostré lokálne extrémy), spustia iba prepočet danej vlastnosti a používateľ uvidí vo výpisoch nové hodnoty.

Posledná záložka s názvom Derivácie ovláda zobrazenie derivácií zadanej funkcie alebo farbu, ktorou sú vykreslené. Nachádza sa tu aj možnosť zmeniť farbu hlavnej funkcie. Zobrazovanie hlavnej funkcie nie je možné vypnúť.

3.4.3 Textové výpisy

Ako bolo spomínané, editor ponúka 2 rôzne typy textových okien. Jeden na textové výpisy a druhý na výpis chýb a upozornení. V tejto časti popíšeme ako samotné výstupy vyzerajú a aký majú formát.



(a) výpisy vypočítaných hodnôt

(b) ukážka chybových hlášok

Obr. 19: Ukážka textových výpisov

Textové výpisy vypočítaných hodnôt

Textové výpisy vypočítaných hodnôt sa vypisujú ako formátovaný reťazec v nasledujúcom tvare:

Dátum a čas vykonania akcie

Názov vlastnosti

Počet nájdených bodov alebo intervalov

Pole x - ových súradníc (v prípade intervalov sú to dvojice bodov (x_1, x_2) kde x_1, x_2 predstavujú krajné body intervalu), v ktorom sú prvky oddelené čiarkami a každá hodnota je na novom riadku.

Ak sa jedná iba o zmenu parametra a výsledkom akcie nie je pole hodnôt, vypíše sa iba názov akcie, názov parametra, ktorý meníme a jeho hodnota.

Textové výpisy chybových hlášok a upozornení

Chybové hlášky majú nasledujúci formát:

Varovanie

Popis chyby

Súbor, kde chyba nastala

Kategória chyby

Táto funkcionalita slúži iba na odchytávanie chýb a ich výpis. Pre bežného používateľa teda nemá veľký význam.

4 Implementácia

V tejto kapitole popíšeme samotnú implementáciu editora. Vychádzali sme z literatúry. [2, 5]

4.1 Organizácia súborov

Výsledný softvér je rozdelený do štyroch priečinkov (Python packages), ktoré obsahujú zdrojové súbory a implementáciu.

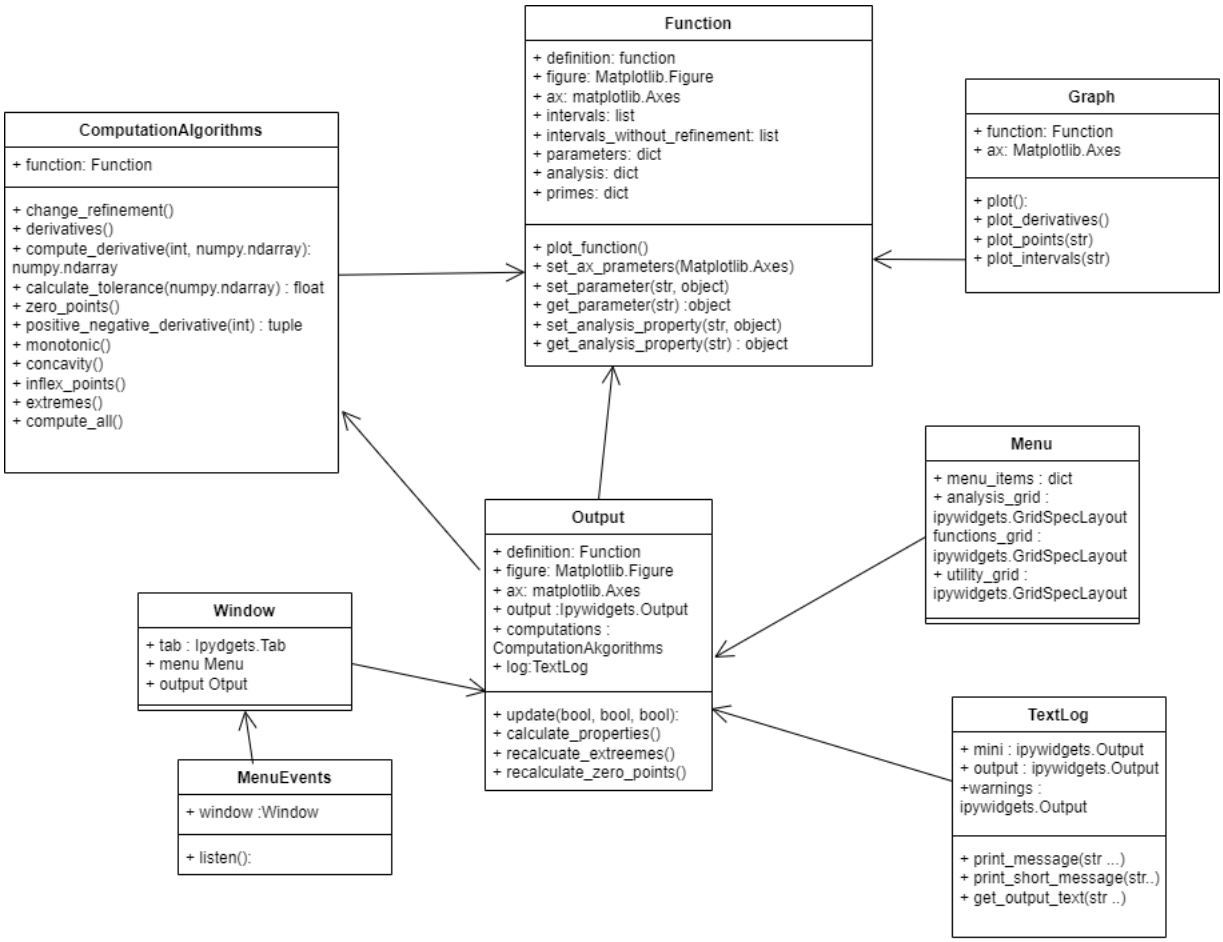
- analysis - Obsahuje reprezentácie funkcie, jej grafu a algoritmy na vyšetrovanie priebehu.
- interactive - Obsahuje metódy potrebné na inicializáciu celého systému.
- gui - Obsahuje implementáciu používateľského rozhrania a jeho ovládanie.
- properties - Číta konfiguračný súbor, v ktorom sú definované predvolené parametre, ktoré editor používa.

4.2 UML Class diagram

V tomto diagrame (Obr. 20) môžeme vidieť jednotlivé triedy s ich atribútmi a dôležitými metódami a asociácie medzi nimi.

4.3 Reprezentácia funkcie a grafu

Výsledný editor bude fungovať v prostredí Jupyter Notebook. To znamená, že používateľ bude väčšinou chcieť zobraziť graf funkcie na jednom obrázku a graficky vyšetriť jeho priebeh na samostatnom obrázku. Graf funkcie, ktorý bude zobrazovať editor, teda vznikne vytvorením nového obrázku a zmena grafu v editore neovplyvní pôvodný graf.



Obr. 20: Triedny diagram

4.3.1 Funkcia

Elementárnu funkciu v našej implementácii reprezentujeme triedou Function. Túto triedu následne využívajú takmer všetky ostatné triedy. Táto trieda pri svojej inicializácii dostane všetky potrebné parametre, ktoré boli zadané používateľom, popísané v časti 3.3 a využívame ich v iných triedach. Function obsahuje dátovú štruktúru **parameters**, kde sú uložené informácie o tom, ktorá vlastnosť je aktuálne zobrazená na grafe, parametre potrebné pre výpočty alebo predvolené parametre definované v súbore `properties.yml` a **analysis** a v ktorej sú uložené vypočítané hodnoty. Trieda obsahuje ešte dve metódy.

set_ax_parameters(ax) Metóda dostane parameter **ax** a nastaví mu parametre súradnicových osí, rovnaké, ako boli zadané používateľom. Tým zabezpečujeme, že parametre (mierka, označenie dielikov, ohraničenie osí) grafu vytvoreného editorom budú rovnaké ako má pôvodný graf.

plot_function(ax) Metóda vytvorí inštanciu triedy Graph a podľa zvolených parametrov zavolá jej metódy na vykreslenie vlastností.

4.3.2 Graf

Graf funkcie reprezentujeme triedou Graph. Táto trieda slúži na vykresľovanie grafu funkcie pomocou knižnice Matplotlib. Obsahuje nasledujúce metódy:

plot() Metóda vykreslí graf hlavnej funkcie.

plot_points(name) Metóda vykreslí na grafe význačné body podľa parametra *name* (zero_points, inflex_points, extremes).

plot_intervals(name) Metóda vykreslí na grafe intervaly podľa parametra *name* (increasing, decreasing, concave_up, concave_down)

plot_derivatives() Metóda vykreslí grafy derivácií, pokiaľ sa ich používateľ rozhodol zobraziť.

4.3.3 Grafické výstupy

Grafické výstupy rieši trieda Output. Táto trieda pri inicializácii vyrobí kópiu objektov potrebných na vykreslenie grafu (Figure, Axis), ktoré potom používa trieda Graph. Táto trieda obsahuje iba jednu metódu, ktorá aktualizuje vykreslený graf na základe akcie z používateľského rozhrania. Grafické výstupy sú zobrazené v editore pomocou wigdetu Output z knižnice Ipywidgets.

update(complete=False, zero_points=False, extremes=False) Metóda zabezpečuje aktualizáciu grafu. Ak je potrebné, tak sa prepočítajú potrebné hodnoty (Parameter complete=True znamená prepočítanie všetkých vlastností). Pri prepočtoch sa tiež odchytávajú prípadné chybové hlášky a varovania, ktoré sa následne vypíšu na textový výstup.

4.3.4 Implemenácia výpočtových algoritmov

Trieda ComputationsAlgorithms obsahuje metódy na výpočet vlastností podľa viet a definícií v časti 2.1 a 2.2. Takmer všetky vety pracujú s nulovými bodmi derivácií. Preto musíme implementovať funkcie, ktoré nám pomôžu ich aproximovať. Všetky nižšie spomínané funkcie sú implementované s pomocou knižnice Numpy, indexovania polí a iných pomocných funkcií.

zero_crossings(array) Funkcia dostane parameter **array**, numpy pole funkčných hodnôt derivácií a vráti dvojice indexov, kde sa mení znamienko.

approximate_zeros(array, atol) Funkcia dostane parameter, numpy pole funkčných hodnôt derivácií a toleranciu a nájde indexy hodnôt blízkych nule. Ak sú hodnoty poľa **array** na daných indexoch blízke nule, (v tolerancii **atol**) hodnoty zmeníme na 0. Následne zmeníme hodnoty na indexoch, kde sa mení znamienko, na 0. Funkcia vráti upravené pole.

change_refinement() Metóda prepočíta jednotlivé intervaly podľa zjemnenia zadaného používateľom. Použije pri tom pôvodné intervaly. Pre každý zadaný interval zistí jeho krajné body a vráti nový interval s požadovaným počtom bodov.

compute_derivative(n, interval) Metóda numericky vypočíta funkčné hodnoty derivácie stupňa **n** pre daný **interval**. Počet bodov použitých na výpočet derivácie závisí od stupňa derivácie a presnosti zadanej používateľom. Metóda vráti pole funkčných hodnôt derivácie.

calculate_tolerance(interval) Metóda vypočíta toleranciu pre daný **interval** podľa mierky grafu a veľkosti intervalu. Táto tolerancia sa používa pri aproximácii nulových bodov derivácií.

zero_points() Metóda vypočíta približné súradnice nulových bodov pomocou implementácie Newtonovej metódy z knižnice Scipy. Newtonova metóda vráti približnú x-ovú súradnicu nulového bodu. Následne skontroluje, či táto hodnota patrí do nami zvoleného intervalu a či je funkčná hodnota v danom bode blízka nule. Ak tento bod spĺňa podmienky, tak je nulovým bodom.

positive_negative_derivative(n) Algoritmus pre zisťovanie intervalov monotónnosti a konvexnosti (konkávnosti) je v podstate rovnaký až na stupeň derivácie, s ktorou pracujeme. (Vychádzame z viet v častiach 2.1.6 a 2.1.7).Táto pomocná metóda rozdelí intervaly podľa toho, kde je derivácia kladná, respektíve záporná. Parameter n určuje stupeň derivácie. Metóda najprv spojí intervaly a funkčné hodnoty derivácií do dvojrozmerného poľa pomocou metódy **column_stack** z knižnice Numpy. Pôvodné polia sa stanú stĺpcami dvojrozmerného poľa. V takto vytvorenom poli vieme jednoducho filtrovať intervaly pomocou hodnôt derivácie. Metóda vráti dvojicu intervalov rozdelených podľa znamienka derivácie.

monotonic() Metóda použije výsledok funkcie **positive_negative_derivative()** pre prvú deriváciu na získanie intervalov monotónnosti.

concavity() Metóda použije výsledok funkcie **positive_negative_derivative()** pre druhú deriváciu na získanie intervalov konvexnosti/konkávnosti.

inflex_points() Metóda vypočíta približné súradnice inflexných bodov. Vychádame z vety z časti 2.1.8. Na zistenie súradníc inflexných bodov spojíme intervaly s hodnotami druhej a tretej derivácie pomocou **column_stack** a nájdeme súradnice inflexných bodov.

```
1 usage
def inflex_points(self):
    inflex_points = []
    prime2 = self.function.get_analysis_property('derivative2')
    prime3 = self.function.get_analysis_property('derivative3')
    intervals = self.function.get_intervals()
    for i in range(len(intervals)):
        atol = self.calculate_tolerance(intervals[i])
        second_der = approximate_zeros(prime2[i], atol)
        third_der = approximate_zeros(prime3[i], atol)
        stack = np.column_stack((intervals[i], second_der, third_der))
        zero_second_derivative = second_der == 0
        filtered = stack[zero_second_derivative]
        nonzero_third_derivative = filtered[:, 2] != 0
        indices = np.flatnonzero(nonzero_third_derivative)
        inflex_points.extend(filtered[indices, 0])
    self.function.set_analysis_property('inflex_points', prepare(inflex_points,
                                                                  self.function.get_parameter('rounding')))
```

Obr. 21: Metóda na hľadanie inflexných bodov

extremes() Metóda vypočíta približné súradnice ostrých lokálnych extrémov. Vychádzame z vety z časti 2.1.5. Znova využívame **column_stack**. Počet derivácií, ktoré kontrolujeme, určuje používateľ.

round_to_n_significant(array, n) Pomocná funkcia zaokrúhli prvky poľa **array** na **n** platných cifier.

4.4 Používateľské rozhranie

Používateľské rozhranie je rozdelené do niekoľkých tried.

4.4.1 Window

Trieda Window zabezpečuje inicializáciu okna editora. Grafické výstupy, ovládanie a textové výstupy. Táto trieda tiež definuje rozloženie prvkov v okne editora.

4.4.2 Menu

Trieda Menu definuje ovládacie prvky používateľského rozhrania. Využíva pritom prvky knižnice Ipywidgets. Používame z nej ToggleButton, Dropdown, IntText, ColorPicker a tzv. ContainterWidgets. Sú to widgety, ktoré v sebe obsahujú ďalšie widgety. Najviac sme používali Tab na rozdelenie rozhrania do záložiek alebo Hbox, aby sme spojili ToggleButton a ColorPicker na ovládanie zobrazenia jednotlivých vlastností.

4.4.3 MenuEvents

Trieda MenuEvents obsahuje metódy, ktoré menia parametre po akcii z používateľského rozhrania. Hlavná metóda triedy **listen()** postupne prejde cez všetky ovládacie prvky. Každý z prvkov z knižnice Ipywidgets má metódu **observe(handler, name)**, kde **handler** je funkcia, ktorá sa má vykonať v prípade, že sa zmení hodnota atribútu **name**. Pri všetkých widgetoch budeme sledovať atribút **value**. Ďalej má trieda MenuEvents množstvo metód, ktoré ovládajú zobrazenie grafu alebo iné parametre. Trieda dostane v konštruktore inštanciu triedy Window, pomocou ktorej máme prístup k ostatným triedam, ktoré potrebujeme pri reakcii na zmeny v používateľskom rozhraní.

4.4.4 TextLog

Trieda TextLog zabezpečuje textové výpisy vypočítaných hodnôt, upozornení alebo zmien parametrov funkcie. Na tieto účely slúžia tri widgety typu Ipywidgets.Output. Na ukladanie a postupné vypisovanie správ sme implementovali jednoduchý zásobník (Stack). Trieda má tri hlavné metódy:

print_short_message(message, **kwargs) Metóda vypíše skrátenú správu do malej záložky, aby informovala používateľa o poslednej vykonanej akcii.

print_message(message, main=False, warning=False, **kwargs) Metóda vypíše správu do jedného alebo viacerých výstupov (hlavné výpisy, upozornenia), podľa toho, ktorý parameter má hodnotu **True**.

get_message_text(message **kwargs) Správy a výsledky ktoré budeme posielať na textový výstup, môžu mať rôznu dĺžku alebo formát. Všetky metódy, ktoré pripravujú textové výpisy, majú premenlivý počet argumentov (**kwargs). Môže sa objaviť kombinácia reťazcov, polí, alebo jednoduchých hodnôt. Preto sme vytvorili funkciu, ktorá spracuje všetky argumenty a vráti formátovaný reťazec so všetkými potrebnými informáciami. Metódy triedy TextLog sa volajú z triedy MenuEvents, keď používateľ vykoná nejakú akciu.

Záver

Cieľom tejto práce bolo navrhnúť a implementovať interaktívny editor na vyšetrovanie priebehu elementárnych funkcií. Editor ponúka možnosť grafického zobrazovania základných vlastností funkcie alebo jej význačných bodov. Editor tiež dokáže prezentovať výsledky v textovej forme. Používateľ má možnosť meniť rôzne parametre, ktoré ovplyvňujú presnosť vypočítaných výsledkov. Editor je implementovaný v programovacom jazyku Python a má jednoduchú inicializáciu aj používateľské rozhranie. Práca s editorom je jednoduchá a intuitívna. Bežný používateľ,ktorý sa s editorom stretne, na jeho používanie nepotrebuje pokročilé znalosti z matematickej analýzy ani programovania.

Naša implementácia dokáže spoľahlivo nájsť a zobraziť vlastnosti aj pre netriviálne elementárne funkcie. Z časových dôvodov sa nepodarilo editor otestovať so študentami, no počas vývoja sme ho priebežne testovali. Editor má všetku požadovanú funkcionalitu a výsledky, ktoré s jeho pomocou získame, sú dostatočne presné pre potreby študentov predmetu Matematická analýza.

Aj keď je náš editor funkčne kompletný, existujú možnosti na jeho vylepšenie. Mohli by sme pridať možnosť exportovania vypočítaných výsledkov do súboru pre ďalšie použitie, zlepšiť presnosť výsledkov pri niektorých funkciách alebo zlepšiť responzivitu systému pri zobrazovaní funkcie na veľmi jemných intervaloch.

Literatúra

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

[2] Michael T. Goodrich, Roberto Tamassia and Michael H. Goldwasser. *Data Structures and Algorithms in Python*. Wiley, 2013.

[3] Zbyněk Kubáček and Ján Valášek. *Cvičenia z matematickej analýzy I*. Bratislava : Univerzita Komenského, 2009.

[4] Tibor Neubrunn and Jozef Vencko. *Matematická analýza I*. Bratislava : Univerzita Komenského, 1992.

[5] Cyrille Rossant. *Learning IPython for Interactive Computing and Data Visualization*. Packt Publishing, 2nd edition, 2015.

[6] Gilbert Strang and Edwin "Jed" Herman. *Calculus Volume 1* . OpenStax, 2016.

[7] Jupyter Project documentation. jupyter.org. `https://docs.jupyter.org/en/latest/index.html` Dostupné: máj 2023.

[8] Jupyter Widgets documentation. `https://ipywidgets.readthedocs.io/en/stable/` Dostupné: máj 2023.

[9] Matplotlib. `https://matplotlib.org/` Dostupné: máj 2023.

[10] Numpy. `https://numpy.org/` Dostupné: máj 2023.

[11] Scipy. `https://scipy.org/` Dostupné: máj 2023.

[12] findiff documentation. `https://findiff.readthedocs.io/en/latest/index.html` Dostupné: máj 2023.

[13] Geogebra graphing calculator. geogebra.org. `https://www.geogebra.org/graphing` Dostupné: máj 2023.

[14] Brilliant: Calculus in a nutshell brilliant.org. `https://brilliant.org/courses/calculus-nutshell/` Dostupné: máj 2023.

Príloha A: obsah elektronickej prílohy

V elektonickej prílohe priloženej k tejto práci sa nachádza zdrojový kôd editora a niekoľko .ipynb súborov (Jupyter notebooks) s príkladmi na vyšetrovanie priebehu elementárnych funkcií.

Príloha B: Používateľská príručka

Táto príloha obsahuje pokyny na inštaláciu a spustenie editora.

1. Nainštalujete si sofvér Miniconda podľa pokynov na stránke

   `https://conda.io/projects/conda/en/latest/user-guide/install/index.html` (Počas vývoja sme používali Minicondu 3 pre Python 3.9)

2. Po inštalácii otvoríte Anaconda Powershell Prompt (miniconda3) alebo termial ak používate Linux

3. V termináli zadáte príkaz `conda update conda`

4. V termináli zadáte príkaz `conda update --all`

5. Vytvoríte si prostredie príkazom `conda create --name [meno_prostredia]`

6. Aktivujete prostredie príkazom `conda activate [meno_prostredia]`

7. Pridáte kanál na inštaláciu balíkov z ktorého budeme inštalovať potrebné knižnice príkazom `conda config --add channels conda-forge`

8. Nainštalujete potrebné knižnice prikazom

   `conda install python tornado=6.1 jupyter numpy matplotlib pyyaml findiff`

   Je veľmi dôležité naiinštalovať knižnicu tornado verzie 6,1. Novšie verzie obsahujú bug v prostredí Jupyter Notebook, ktorý by znemožnil používanie editora. (V čase odovzdávania práce tento bug ešte nebol odstránený)

9. Spustíte príkaz `jupyter notebook`

10. Spustí sa webový prehliadač a následne môžete prejsť do priečinku kde sa nachádzajú testovacie príklady