Numerický výpočet Simpsonova integrálu se zadanou přesností

Úvod do programování

David Martínek

Geografie a kartografie, 3. ročník

13.02.2022

Zadání úlohy

Na vstupu je spojitá a diferencovatelná funkce v tvaru F(x), interval <a,b>, krok dělení n. Proveďte přibližný výpočet určitého integrálu F(x) dx na intervalu <a,b>, a,b představují meze určitého integrálu.

Návod: Se zadaným krokem n spočtěte na intervalu <a,b> plochu funkce f pod křivkou s využitím Simpsonovy metody. K otestování použijte 3 funkce, hodnoty a, b, n jsou voleny uživatelem.

Součástí odevzdání bude:

- Dokumentace se zadáním (4–5 stran)
    - Rozbor problému
    - Existující algoritmy
    - Popis zvoleného algoritmu
    - Struktura programu
    - Problematická místa
    - Možná vylepšení
- Zdrojový kód aplikace
- Vstupy/výstupy

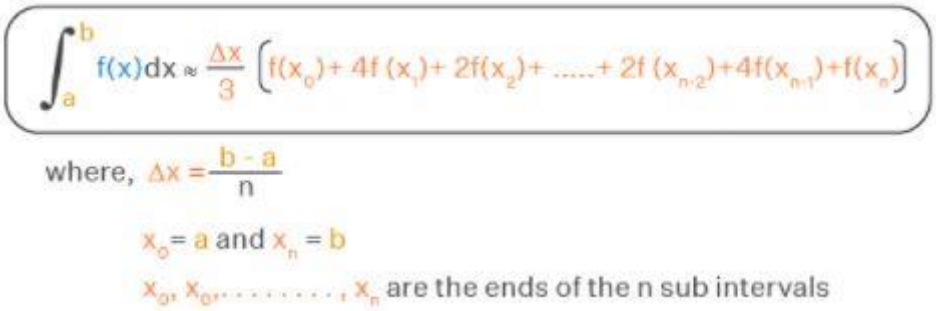
Za nefunkční bude aplikace považována, pokud:

- Při zpracování dat dojde k pádu
- Vrací špatné výsledky
- Neřeší možné singulární případy

Rozbor problému

Při řešení určitého integrálu standardním způsobem se hledá primitivní funkce k funkci integrované. Odečtením od sebe funkčních hodnot mezí primitivní funkce lze následně zjistit hodnotu plochy pod křivkou. Simpsonova metoda nabízí jejímu uživateli obejít integrační fázi cestou aproximace výsledku prostřednictvím Simpsonova vzorce (Cuemath, 2022a):

Obrázek 1 - Simpsonova aproximace integrálu



Zdroj: Cuemath (2022b)

Funkce musí být spojitá a diferencovatelná na celém definičním oboru.

Existující algoritmy

Na rozdíl od tohoto řešení Simpsonovy metody, většina existujících algoritmů využívá k jejímu řešení přímo nadefinovanou funkci. Na portálu Mathematical Python od autorů Walls a kol. (2020a) si však vystačili jen s krátkým zápisem s pomocí knihovny *numpy*:

Obrázek 2 - Existující algoritmus PNM

```
import numpy as np

a = 0
b = np.pi
n = 11
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)
f = np.sin(x)

I_simp = (h/3) * (f[0] + 2*sum(f[:n-2:2]) \
           + 4*sum(f[1:n-1:2]) + f[n-1])
err_simp = 2 - I_simp

print(I_simp)
print(err_simp)
```

Zdroj: Walls a kol. (2020b)

Přímo nadefinovaná funkce *integrate.simpson* se nachází v knihovně *scipy* a může do ní vstupovat až pět parametrů. Parametr *y* je jediným povinným parametrem pro fungování této funkce a popisu pole, které má být integrováno (SciPy, 2022).

Popis zvoleného algoritmu

K nalezení sumy hodnoty všech uzlů (x_n) slouží *for* cyklus o délce *n+1* (kde *n* je počet podintervalů). Algoritmus nejprve podmínkami zjistí, jakou konstantou k má násobit příslušný uzel a následně spočte funkční hodnotu uzlu. Obě hodnoty se vynásobí a takto získané hodnoty v rámci každého průchodu cyklem se sčítají v proměnné *sum*. Ta je po skončení cyklu (jakmile je započítána hodnota každého uzlu) vynásobena hodnotou *h/3*, čímž je získán výsledek celé aproximace.

Pseudokód zvoleného algoritmu:

cyklus pro spočítání každého uzlu

    pokud je uzel v pořadí lichý,

        k = 4

    jinak

        k = 2

    pokud je uzel první nebo poslední v pořadí

        k = 1

    x = součin konstanty h a pořadí uzlu i (včetně nultého bodu)

    hodnota uzlu = k*F(x)

    přičtení hodnoty uzlu k **sumě** hodnot

plocha = (h/3)*suma hodnot

Struktura programu

Program je tvořen 56 řádky včetně komentářů a mezer.

Na začátku programu je uživatel otázán na hodnoty mezí integrálu *a, b*. Pokud uživatel nezadá pro některou z hodnot celé číslo, program ho na to upozorní a spadne. Program pokračuje i v případě, kdy je hodnota dolní meze větší než hodnota horní meze. K zajištění korektního přepočtu slouží proměnné *c* a *d*. Jejich úlohou je meze prohodit a na závěr zajistí převedení výsledku na opačnou hodnotu. Vstup *n* (počet subintervalů) uživatele nepustí v programu dále, dokud mu není nastavena celočíselná kladná hodnota – k tomu je využito nekonečného cyklu.

Po výpočtu hodnoty *h* následuje *for* cyklus, jehož funkčnost je popsána v kapitole popis zvoleného algoritmu.

Na závěr program dopočítá aproximaci a vytiskne hodnotu zjištěné plochy.

Problematická místa

Dokud nebyla situace ošetřena, program si neuměl poradit se situací, ve které byla horní mez integrálu menší než mez dolní. Nyní program pracuje i s touto variantou. Ačkoliv celý program je zatížen o několik řádek navíc, dá se to považovat za lepší řešení, než aby byla uživateli omezována volba vstupních parametrů.

Významným teoretickým problémem celého programu je fakt, že některé zdroje uvádějí nemožnost využití Simpsonovy metody, pokud počet subintervalů *n* není sudý. Pátrání po příčině těchto tvrzení bylo neúspěšné, přesto při testování programu nebyl objeven důvod, proč by lichý počet subintervalů měl s funkcí metody a programu interferovat. Ba naopak při zadání lichého počtu subintervalů byl výsledek aproximace standardně přesnější než při sudém počtu subintervalů.

Testování programu

Program byl testován na tři funkce ve třech podmínkách:

Tabulka 1 - Výsledky testování funkcí

(a ,b, n)	x+1		sin(x)		e^x	
	Program	WolframAlpha	Program	WolframAlpha	Program	WolframAlpha
(0, 5, 5)	17	17.5	0.54	0.72	147.66	147.41
(5, 0, 99)	-17.48	-17.5	-0.72	-0.72	-147.40	-147.41
(10,11,999)	11.5	11.5	-0.84	-0.84	37840.32	37848

Zdroj: vlastní tvorba s využitím dat z portálu WolframAlpha (2022)

Výsledky testování ukazují, že přesnější aproximace je docíleno s použitím většího množství subintervalů. Testy zároveň prokázaly, že i s větší dolní než horní mezí se program umí dobrat správného výsledku. Všechny použité funkce jsou spojité a diferencovatelné na celém definičním oboru. Program by si nedokázal poradit s nespojitou funkcí.

Možná vylepšení

Program by mohl při jeho otevření nabízet uživateli sestrojení funkce, kterou potřebuje integrovat. Zároveň by program mohl vypočítat přibližnou chybu aproximace, aby uživatele upozorňoval na použití nedostatečného množství subintervalů.

Pro náročné uživatele by program mohl vykreslovat graf popisující situaci pod křivkou v zadaných mezích, ba dokonce ilustrovat rozčlenění osy x na uzly a subintervaly.

Zdroje

Cuemath (2022): Cuemath, Simpson's Rule,
https://www.cuemath.com/simpsons-rule-formula/

P. Walls, I. Hawke, Daniel, W. Lai (2020): Mathematical Python, Simpson's Rule,
https://personal.math.ubc.ca/~pwalls/math-python/integration/simpsons-rule/

SciPy (2022): SciPy, Integrate Simpson,
https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.simpson.html

WolframAlpha (2022): WolframAlpha, Computional intelligence,
https://www.wolframalpha.com/