УДК 004.42:519.6

В. В. Химич
О. І. Тютюнник

АВТОМАТИЗАЦІЯ СИСТЕМИ АНАЛІТИЧНОГО ОБЧИСЛЕННЯ ВИЗНАЧЕНИХ ТА НЕВИЗНАЧЕНИХ ІНТЕГРАЛІВ

Вінницький національний технічний університет

Анотація

У роботі розглядається задача розробки та програмної реалізації застосунку для символьного (аналітичного) та чисельного обчислення математичних виразів, зокрема визначених та невизначених інтегралів. Об'єктом дослідження є алгоритми комп'ютерної алгебри та методи побудови графічних інтерфейсів користувача (GUI). У ході роботи розроблено програмне забезпечення мовою Python із використанням бібліотек Tkinter для візуалізації та SymPy для математичного ядра. Проведено тестування системи на різних класах функцій (поліноміальні, тригонометричні, логарифмічні), а також реалізовано алгоритм автоматичного переходу до чисельного інтегрування для функцій, що не мають первісної в елементарних функціях (наприклад, інтеграли Френеля). Результати підтверджують високу точність обчислень та ергономічність розробленого візуального редактора формул.

Ключові слова: комп'ютерна алгебра, інтегрування, графічний інтерфейс, Tkinter, SymPy, символьні обчислення, Python.

Abstract

This paper addresses the problem of developing and implementing an application for symbolic (analytical) and numerical computation of mathematical expressions, specifically definite and indefinite integrals. The object of the study is computer algebra algorithms and methods for building graphical user interfaces (GUI). During the work, software was developed in Python using the Tkinter library for visualization and SymPy for the mathematical core. The system was tested on various classes of functions (polynomial, trigonometric, logarithmic), and an algorithm for automatic transition to numerical integration for functions without elementary antiderivatives (e.g., Fresnel integrals) was implemented. The results confirm the high accuracy of the calculations and the ergonomics of the developed visual formula editor.

Keywords: computer algebra, integration, graphical interface, Tkinter, SymPy, symbolic computations, Python.

Вступ

Розвиток систем комп'ютерної алгебри значно спростив процес розв'язання складних математичних задач в інженерній та науковій практиці. Символьне інтегрування є однією з базових і водночас найбільш ресурсоємних задач математичного аналізу. Більшість існуючих потужних математичних пакетів є або комерційними продуктами з високим порогом входження, або вимагають постійного підключення до мережі Інтернет [1,2].

Дана робота присвячена розробці локального, незалежного від інтернет-з'єднання програмного засобу для аналітичного обчислення інтегралів. Основна увага приділяється не лише інтеграції потужного математичного ядра SymPy для точних символьних обчислень, але й створенню інтуїтивно зрозумілого графічного візуального редактора мовою Python (Tkinter) [4,5]. Запропонований редактор дозволяє користувачу конструювати математичні вирази (дроби, корені, межі інтегрування) за допомогою спеціалізованих віджетів, що мінімізує синтаксичні помилки вводу. Комплексний підхід до обробки даних включає попередній парсинг візуальних структур у машинозрозумілий формат, застосування регулярних виразів для усунення неоднозначностей (наприклад, неявного множення) та автоматичний перехід до методів наближеного обчислення у випадках відсутності аналітичного розв'язку.

Результати досліджень

Оскільки розроблений програмний комплекс має розгалужену архітектуру, для детального розгляду виділено три ключові модулі, що забезпечують його функціонування: підсистему візуального вводу (система «слотів»), модульну структуру математичних виразів та ядро символьних обчислень.

1. Розробка підсистеми візуального вводу (клас Slot). Для усунення проблеми синтаксичних помилок, які часто виникають при класичному рядковому введенні формул, було розроблено концепцію «візуального слота». Слот – це інтерактивний елемент на базі Tkinter.Canvas, який динамічно адаптується під введений текст.

```
class Slot(tk.Canvas):
    def __init__(self, parent, min_chars=3, font=None, on_change=None, small=False):
        # Ініціалізація розмірів та шрифтів
        char_w = 11 if small else 14
        h = 28 if small else 36
        w = max(min_chars * char_w + 12, 36)
        super().__init__(parent, width=w, height=h, bg=SLOT_BG, bd=0)
        self._var = tk.StringVar()
        self._var.trace_add("write", self._on_text_change)
        self._entry = tk.Entry(self, textvariable=self._var, font=self._font)
        # ... налаштування фокусу та клавіатурної навігації
```

Клас інкапсулює стандартний віджет tk.Entry всередині полотна. Завдяки методу _on_text_change(), який відслідковує зміну змінної _var, ширина слота автоматично перераховується залежно від кількості введених символів. Крім того, реалізовано внутрішній реєстр слотів (_registry) для безшовного переміщення курсору між різними частинами формули за допомогою клавіатурних стрілок.

2. Архітектура візуального математичного блоку інтеграла. Математичний вираз у програмі будується з незалежних блоків (дроби, корені, степені). Найскладнішим елементом є блок інтеграла (IntegralBlock), який агрегує чотири окремі слоти: верхню межу, нижню межу, підінтегральний вираз та диференціал змінної.

```
class IntegralBlock(tk.Frame):
    def __init__(self, parent, chg):
        super().__init__(parent, bg=BG)
        # Створення слотів для меж інтегрування
        self.upper = Slot(lim_col, min_chars=3, small=True, on_change=chg)
        self.lower = Slot(lim_col, min_chars=3, small=True, on_change=chg)
        # Слот для підінтегральної функції
        self.integrand = Slot(mid, min_chars=6, on_change=chg)
        # Слот для змінної інтегрування (наприклад, dx)
        self.var = Slot(d_col, min_chars=1, on_change=chg)
    def to_sympy_str(self):
        lo, hi = self.lower.get().strip(), self.upper.get().strip()
        ig     = self.integrand.get().strip()
        v      = self.var.get().strip() or "x"
        return lo, hi, ig, v
```

Цей клас не лише відповідає за правильне геометричне позиціонування елементів (знак інтеграла ∫, межі над і під ним), але й виконує функцію збору даних. Метод to_sympy_str() витягує рядкові значення з усіх чотирьох слотів і передає їх математичному ядру для подальшого парсингу.

3. Ядро аналітичних та чисельних обчислень. Основою обчислювальної системи є функція compute_integral, яка є своєрідним "мостом" між візуальним інтерфейсом Tkinter та математичним рушієм SymPy.

```
def compute_integral(line):
    # ... отримання даних з блоку інтеграла (lo, hi, ig, v)
```

```
    try:
        def clean(s):
            if not s: return ""
            # Санітизація візуальних символів у машинозрозумілий код
            return s.replace("·", "*").replace("−", "-").replace("^", "**").replace("π", "pi").replace("∞",
"oo").replace("√", "sqrt")
        # Парсинг змінної та виразу у формат AST (Abstract Syntax Tree)
        sym = sympy.Symbol(clean(v) or "x")
        expr = sympy.sympify(clean(ig), locals={str(sym): sym, "pi": sympy.pi, "e": sympy.E, "oo": sympy.oo,
"sqrt": sympy.sqrt})
        # Обчислення (Визначений або Невизначений інтеграл)
        if lo and hi:
            lo_v = sympy.sympify(clean(lo), locals={"pi": sympy.pi, "e": sympy.E, "oo": sympy.oo})
            hi_v = sympy.sympify(clean(hi), locals={"pi": sympy.pi, "e": sympy.E, "oo": sympy.oo})
            result = sympy.integrate(expr, (sym, lo_v, hi_v))
        else:
            result = sympy.integrate(expr, sym)
        # Автоматичний перехід до чисельного наближення для нерозв'язних інтегралів
        if hasattr(result, 'has'):
            if result.has(sympy.Integral) or 'fresnel' in str(result).lower():
                result = result.evalf(6)
                simp_res = sympy.expand(sympy.simplify(result))
# ... форматування результату для виводу
```

Робота ядра поділяється на кілька етапів. Перший етап – лексичний аналіз (функція clean), який замінює візуально красиві символи (наприклад, "·" або "√") на оператори, зрозумілі Python ("*" та "sqrt"). Другий етап – побудова математичної моделі за допомогою sympy.sympify(), що перетворює рядок тексту на абстрактне синтаксичне дерево. Третій етап – безпосередньо інтегрування за допомогою sympy.integrate(). Важливою особливістю розробленого алгоритму є наявність перевірки результату: якщо аналітичний розв’язок відсутній (наприклад, з'являються інтеграли Френеля), система автоматично викликає метод evalf(6), переходячи від символьного до чисельного інтегрування з точністю до шести знаків після коми.

Висновки

У роботі спроєктовано та програмно реалізовано автоматизований комплекс для аналітичного та чисельного обчислення математичних інтегралів мовою Python. Важливим досягненням стала розробка ергономічного графічного інтерфейсу на базі підходу «візуального конструктора», де використання окремих слотів для введення виразів і меж інтегрування разом із механізмом інтелектуального парсингу мінімізувало ймовірність синтаксичних помилок та наблизило взаємодію з програмою до класичного математичного запису. Інтеграція математичного ядра SymPy забезпечила високу точність символьних перетворень для різних класів функцій, а завдяки функції ідентифікації інтегралів, що не мають розв’язку в елементарних функціях, система здатна автоматично переходити до методів чисельного наближення. Створений продукт продемонстрував стабільність роботи та алгоритмічну гнучкість, що дозволяє ефективно використовувати його як допоміжний електронний інструментарій у навчальному процесі для самоперевірки студентів та дослідження складних інтегральних функцій.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Крак Ю. В. «Алгоритми комп’ютерної алгебри та їх застосування»: навчальний посібник. — Київ: КНУ ім. Т. Шевченка, 2022.
2. Дзюба Т. А. «Чисельні методи та алгоритми комп’ютерної математики»: практикум. — Харків: ХНУРЕ, 2021.

3. Meurer A., Smith C. P., Paprocki M. et al. SymPy: symbolic computing in Python // PeerJ Computer Science. — 2017. — Vol. 3. — P. e103.
4. Трофименко О. Г., Прокоп Ю. В., Логінова Н. І. Програмування мовою Python: навчальний посібник. — Одеса: Фенікс, 2021. — 272 с.
5. Tkinter — Python interface to Tcl/Tk [Електронний ресурс]. — Режим доступу: https://docs.python.org/3/library/tkinter.html.

Химич Вячеслав Вадимович **–** студент групи 2 БС – 25б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: 04-25-002.stud@vntu.edu.ua

Тютюнник Оксана Іванівна – доцент кафедри вищої математики, Вінницький національний технічний університет, м. Вінниця, e-mail: tyutyunnik@vntu.edu.ua

Chymych Viacheslav V. – student of Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: 04-25-002.stud@vntu.edu.ua

Tiutiunnyk Oksana I. – Associate Professor, Department of Higher Mathematics, Vinnytsia National Technical University, Vinnytsia, e-mail: tyutyunnik@vntu.edu.ua